TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

KALLE BECKER

HITSAUSSAUMAN PAIKAN MITTAUSMENETELMÄN KEHITTÄMINEN LAADUNVARMISTUSAUTOMAATIOON

Diplomityö

Tarkastaja: professori Risto Ritala

TIIVISTELMÄ

Kalle Becker: Hitsaussauman paikan mittausmenetelmän kehittäminen laadunvarmistusautomaatioon
Tampereen teknillinen yliopisto
Diplomityö, 34 sivua, 18 liitesivua
Joulukuu 2015
Systeemitekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Mittaustekniikka
Tarkastaja: professori Risto Ritala

Avainsanat: Konenäkö, ohjelmistokehitys, CAD ohjelmointi, fotogrammetria

Autoteollisuus valmistaa tuotteisiinsa osia, joiden pintamuotoa on vaikeaa ja tehotonta tarkistaa täydellisesti ilman konenäköä. Työssä käytettyä kehitettävän ohjelmiston konenäköä käytetään autoteollisuudessa osien muodon tarkastamiseen ja dokumentointiin. Jotta järjestelmä pystyy tarkistamaan kappaleiden hitsaussaumat, tulisi ne määrittää kuvadatasta manuaalisesti. Sarjatuotantokappaleissa tämä aiheutti turhan suuren työmäärän ja samalla hidasti ohjelman käytettävyyttä. Parantamalla ohjelman toiminnallisuutta piirteiden hitsaussaumojen määrittämisessä voitiin potentiaalisesti tehostaa laaduntarkastusta sekä poistaa hitsaussauman manuaalisessa määrityksessä tapahtuvat virheet.

Diplomityössä ratkaistavana ongelmana oli luoda automaattinen määritys hitsaussaumoille ja samalla vähentää ohjelman käyttöä edellyttävän manuaalisten työn osuutta. Samalla pystyttiin tutkimaan onko automaattisessa määrityksessä käytettävä transformointi soveltuva laadunmittaukseen. Ohjelmistokehityksen tulosta pystyttiin testaamaan valmiilla testidatalla ilman että fyysistä mittausympäristöä tarvitsi käyttää datan hankkimiseksi.

Tulokseksi saatiin lisättyä ohjelmaan automaattinen määritys hitsaussaumoille sekä testattua määritystä tekevän ohjelman toimivuutta eri hitsaussaumojen pistedatalla. Hitsaussaumoille tehtävä globaali transformointi ei tässä konenäköympäristössä saatavan datan vähyyden takia ole riittävän hyvä laaduntarkastukseen. Saumojen paremmaksi automaattiseksi määrittämiseksi tulisi kehittää saatavilla olevan datan laatua tai suorittaa hitsaussaumojen transformointi lokaalimmin.

ABSTRACT

Kalle Becker: Development of welding seam position measurement method for quality assurance automation
Tampere University of Technology
Master of Science Thesis, 34 pages, 18 appendixes
December 2015
Major: Automation Science and Engineering
Examiner: Professor Risto Ritala

Keywords: Machine vision, software development, CAD-based programming, photogrammetry

The card industry uses machine vision in increasing amounts to perform quality inspections on their products. The commercial software used in the making of this thesis is used in the in-line inspection process of different car manufacturers. The software is used to analyze and document data obtained by a machine vision system. This thesis focused on developing the inspection of welding seams by automating the work flow used in defining welding seams from images. The definition had previously created an unwanted amount of work for the end user and slowed down the usability of the software. By improving the functionality of the software it was possible to partly improve the process for quality assurance and to minimize human error created in the defining of welding seams.

In this thesis the goal was to create an automatic segmentation tool for weld seams and at the same time lower the amount of manual adjustment required to run the software. Through the implementation of the new functionality we were able to research the sufficiency of the transformation method used on the weld seams while considering the requirements of quality assurance. The new functionality was tested by using documented measurement data to avoid the need to use any physical measurement system.

Results include examples obtained using the created tool. The used transformation for welding seams was insufficient considering the quality of point data that is available for the current software. To automatically define weld seams in the software the quality of data has to be improved or the transformation needs to be performed locally.

ALKUSANAT

Tämä diplomityö on tehty Hervannan VTT:n toimitiloissa vuoden 2015 aikana. Työ tehtiin osana FIMECC SHOK:in Manu-ohjelmaa ja sen tarkoitus on toimia oppaana miten konenäköä voidaan soveltaa laaduntarkastukseen erityisolosuhteissa.

Haluan kiittää erityisesti työn tarkastajana toiminutta professori Risto Ritalaa Tampereen teknillisestä yliopistosta, sekä ohjaajana toiminutta Matti Kutilaa VTT:ltä. Heille esitän kiitokset asiantuntevista neuvoista työn suunnittelemiseksi ja rajaamiseksi. Suurkiitokset kuuluvat myös Antti Knuutilalle, Kosti Kannakselle sekä Jarkko Leinolle heidän kärsivällisyydestään ja opastuksestaan työn tekemiseksi ja dokumentoimiseksi.

Tampereella 28.12.2015

Kalle Becker

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

.3d	Mapvision ohjelmiston tiedostotyyppi kolmiulotteiselle datalle.
CAD	Computer Assisted Design. Tietokoneavusteinen suunnittelu.
Globaali transformaatio	Pisteiden siirtäminen koordinaatistosta toiseen globaaleja koordinaatteja käyttäen
IDE	Integrated Development Environment. Ohjelmistokehity-sympäristö.
Kiinniottopiste	Transformointiin käytettävä kappaleen piste, yleensä reikäpiirre.
Lokaali transformaatio	Pisteiden siirtäminen koordinaatistosta toiseen paikallisia koordinaatteja käyttäen
Pattern	Työssä käytettävä 2-D olio.
Pistepilvi	Joukko pisteitä sijoitettuna koordinaatistoon
Segment	Pattern olioista ja kuvadatasta koostuva olio.
Siemens NX	Kaupallinen CAD suunnitteluun luotu ohjelmisto.
XML	Extensible Markup Language. Työssä käytettävän pistedatan dokumenttityyppi.
ZB	Zusammenbau, kokoonpano
UZB	Untenzusammenbau, alikokoonpano

1. JOHDANTO

Työn kohteena on autoteollisuuden konenäköön perustuva laadunvalvonta, tarkemmin rajattuna hitsaussaumoille käytetyn laadunvalvonnan kehittäminen. Työ tehtiin muokkaamalla kaupallisen konenäköjärjestelmän ohjelmistoa lisäämällä siihen uudet toiminnallisuudet.

1.1 Konenäön käyttäminen Autoteollisuudessa

Konenäkö on ollut laajassa käytössä eri teollisuuden aloilla jo monia vuosia. Aiemmin konenäkösovelluksien luominen on vaatinut korkeampaa teknistä osaamista, mutta nykyiset sovellukset ovat tehokkaampia ja helppokäyttöisempiä johtuen useiden liitännäisten teknologioiden kehityksestä. Konenäöstä on tullut tärkeä ja joustava työkalu osavalmistukseen niin pienteollisuudelle kuin suurille alkuperäisille laitevalmistajille [1].

Kasvava monimutkaisuus nykyaikaisten autojen valmistuksessa lisää auton osien valmistusvirheiden mahdollisuutta. Valmistajien on minimoitava lopputuotteissa esiintyviä virheitä säilyttääkseen kilpailukykynsä markkinoilla. Tiukkojen laatuvaatimusten ansiosta osatoimittajat ja autotehtaat ovat alkaneet investoimaan virheet havaitseviin konenäköratkaisuihin. Autoteollisuus käyttää konenäköä pääosin laaduntarkkailuun tuotannon eri vaiheissa sekä robottien ohjaukseen.

Konenäköä käytetään löytämään haluttuja piirteitä yhden tai useamman kameran ottamasta kuvasta. Konenäöstä saatavan tiedon avulla robotit poimivat osia hyllyistä, siirtävät laatikoita tai asemoivat osia kokoamisprosessia varten.

Nykyaikaiset osien laatutarkastuksissa käytettävät konenäköjärjestelmät tarkastavat tehokkaasti osien koko- ja muototoleranssit. Konenäköjärjestelmillä voidaan myös tarkastaa kappaleen pintamuodot kosmeettisten virheiden löytämiseksi tai havaita toiminnalliset virheet kuten muotopoikkeamat ja vauriot auton peltiosissa. Konenäöstä saatavia tietoja voidaan myös käyttää myös alkuperäisen tarkastuksen jälkeen esimerkiksi reklamaatiotapauksissa.

Autoteollisuuden tuotantolinjalla osatarkastukset on aikaisemmin suoritettu mittaamalla tasaisin välein valittu näytekappale. Kappaleen dimensiot on mitattu mekaanisella mittapäällä ja mittauksen tulokset taulukoitu. Kun tätä dataa oli kerätty tarpeeksi käytettiin tilastollisia menetelmiä piirtämään trendiviivoja, joita seuraamalla tehtiin päätöksiä valmistusprosessiin tehtävistä muutoksista. Muutoksien hitauden takia syntyi kehitystarve kustannustehokkaammalle mittaamiselle laadun takaamiseksi.

Selvimmät muutokset laadunmittausprosessiin ovat syntyneet konenäköä hyödyntävien ”on-line” mittausten myötä. Konenäkö voidaan asentaa haluttuun kohtaan tuotantolinjalle. Tarkastuksen läpi kulkevan osan tai tuotteen tarkastamisaika on huomattavasti lyhyempi verrattuna manuaalisiin menetelmiin. Suoraan tuotantolinjalta saatavaa tietoa voidaan hyödyntää tilastollisessa prosessinohjauksessa.

1.2 Työn tavoite

Työssä tuli kehittää autoteollisuudessa käytetyn konenäköjärjestelmän ohjelmistoa. Ohjelmistolla ohjataan konenäköjärjestelmää ja käsitellään siitä saatavaa dataa. Tavoitteena oli luoda toiminnallisuus hitsaussaumojen paikan automaattiseksi määrittämiseksi. Samalla pyrittiin vähentämään ohjelman käytössä vaaditun manuaalisen työn osuutta.

Kehityksen kohteena oli kaupallinen ohjelmisto, jonka lähdekoodi on kirjoitettu käyttäen C, C++ ja C# kieliä. Ohjelmistossa suoritettava hitsaussaumojen määrittäminen vaati vastinpisteiden manuaalista poimintaa, mikä haluttiin automatisoida. Lopputuloksen tarkkuus riippuu vastinpisteille tehtävän globaalin transformoinnin riittävyydestä. Globaalilla transformoinnilla tarkoitetaan 3-D pisteille tehtyä globaalia koordinaattimuunnosta.

Ohjelman uuden osuuden testaaminen onnistui valmiin kuvadatan, kamerakalibroinnin ja 3-D mallin avulla. Ohjelmiston testaaminen voitiin toteuttaa yksinkertaisesti käyttämällä valmiita mittausdatoja ja kalibrointitietoja sen sijaan että ohjelmaa oltaisiin ajettu konenäön kanssa fyysisessä mittausympäristössä.

1.3 Työn rakenne

Tämän työn toisessa luvussa esitetään työn menetelmien teoriaa kirjallisuuden perusteella sekä määritetään työn kannalta keskeisiä käsitteitä.

Neljännessä luvussa käsitellään työn suunnitteluvaiheet ja tehdään luotavalle ohjelmalle vaatimusmäärittely.

Luvussa 5 käydään läpi kuvaavasti ohjelmoinnin työvaiheet, sekä selvennetään ohjelman valmiita toiminnallisuuksia.

Luvussa 6 esitellään ohjelmalla ajettu testidata sekä analysoidaan tästä saadut tulokset ja ilmenneet kehitystarpeet.

2. KONENÄÖN PERIAATTEITA

Käytetään kahta referenssikoordinaatistoa, k*uvakoordinaatistoa* ja *kohdekoordinaatistoa*. Kuvakoordinaatisto on kameran kuvatason 2-D koordinaatisto, jonka origo on kuvan vasemmassa yläkulmassa. Pisteen sijainti kuvalla ilmoitetaan vasenkätisenä sarake- rivi koordinaattina.

Kohdeavaruus on kuvan sisältämä kolmiulotteinen näkymä. *Kohdekoordinaatisto* on suorakulmainen koordinaatisto, jota käytetään paikantamaan pisteitä kohdeavaruudessa. Kohdekoordinaatiston origona toimii sopiva paikallinen piste.

Työn kannalta oleellisimmat geometriset laskuoperaatiot ovat pisteavaruuden siirtäminen koordinaatistosta toiseen transformaatiomatriisilla sekä eteen- ja taaksepäinleikkaus.

2.1 Ulkoinen ja sisäinen orientaatio

Sisäisellä orientaatiolla esitetään sensorin tai kameran ominaisuudet, joita tarvitaan kohteesta saatavan kuvan muodostamiseen. Sisäisen orientaation parametrit määritetään kameran kalibrointivaiheessa. Kameran sisäisen orientaation parametrit sisältävät linssin polttovälin, kuvan keskipisteen sijainnin kuvatasossa sekä tiedot linssivääristymistä. Sisäisellä orientoinnilla selvitetään minkälainen projektiosädekimppu muodostaa kuvan.

Ulkoinen orientaatio määrittää sädekimpun sijainnin ja orientaation kohdeavaruudessa. Sädekimpun projektiokeskuksen *L* koordinaatit kohdeavaruudessa määritetään:

$$L = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \tag{1}$$

Ulkoinen orientaatio määrittää kohde- ja kuva-avaruuden koordinaattien välisen suhteen yleistä muotoa olevalla yhtälöllä:

$$\begin{bmatrix} x \\ y \\ -f \end{bmatrix} = k\boldsymbol{M} \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix}, \boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{2}$$

missä (*x, y, -f*) ovat kuva avaruuden koordinaatit, *k* on skaalauskerroin, $\boldsymbol{M}$ on rotaatioparametrit sisältävä 3 x 3 rotaatiomatriisi. Rotaatiomatriisi $\boldsymbol{M}$ rakentuu eri akseleiden ympäri tehtävien rotaatiomatriisien tulosta [3]. Koordinaatit (*X, Y, Z*) ovat kohdepisteen koordinaatit ja (X_0, Y_0, Z_0) kuvan sijainti.

Yleinen lähestymistapa rotaatiomatriisin rakentamiseksi on käyttää kolmea peräkkäistä rotaatiota: ω rotaatio X-akselin ympäri, ϕ rotaatio kererran käännetyn Y-akselin ympäri ja κ rotaatio kaksi kertaa käännetyn Z-akselin ympäri. Matriisit näille rotaatioille ovat:

$$\boldsymbol{M}_{\omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\omega & sin\omega \\ 0 & -sin\omega & cos\omega \end{bmatrix} \tag{3}$$

$$\boldsymbol{M}_{\phi} = \begin{bmatrix} cos\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{bmatrix} \tag{4}$$

$$\boldsymbol{M}_{\kappa} = \begin{bmatrix} cos\kappa & sin\kappa & 0 \\ -sin\kappa & cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

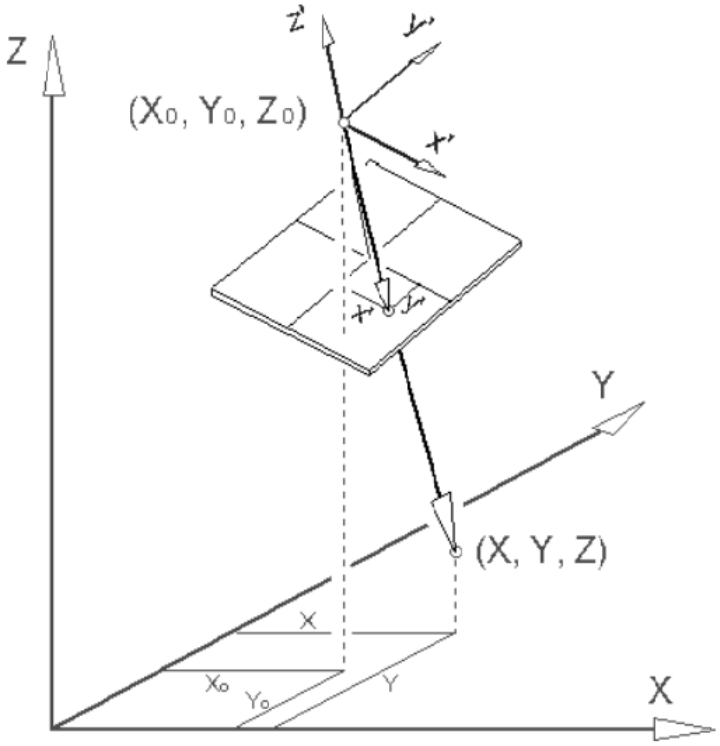
Täydellinen rotaatiomatriisi saadaan näiiden matriisien tulona:

$$\boldsymbol{M} = \boldsymbol{M}_{\kappa}\boldsymbol{M}_{\phi}\boldsymbol{M}_{\omega} \tag{6}$$

$$\boldsymbol{M} = \begin{bmatrix} cos\phi cos\kappa & cos\omega sin\kappa + sin\omega sin\phi cos\kappa & sin\omega sin\kappa - cos\omega sin\phi cos\kappa \\ -cos\phi sin\kappa & cos\omega cos\kappa - sin\omega sin\phi sin\kappa & sin\omega cos\kappa + cos\omega sin\phi sin\kappa \\ sin\phi & -sin\omega cos\phi & cos\omega cos\kappa \end{bmatrix}$$

2.2 Kollineaarisuusehto

Kollineariisuus- eli samasuoraisuusehdon toteutuminen edellyttää kuvahavainnosta muodostetun sädekimpun olevan yhdenmuotoinen sen sädekimpun kanssa, joka kuvatessa projisoituu kohteesta kameraan. Kameran projektiokeskuksen, kuvapisteen ja kohdepisteen tulee siis sijaita samalla suoralla. Oletetaan, että kamera- ja kohdekoordinaatit esitetään ortonormeeratuissa koordinaatistoissa. Tämä tarkoittaa sitä että koordinaattiakselit sijaitsevat suorakulmaisesti toistensa suhteen ja niidenyksikkövektorit ovat samansuuruiset.



Kuva 1. *Musta havaintovektori, joka lähtee projektiokeskuksesta* (X_0, Y_0, Z_0)*, kulkee kuvapisteen läpi ja jatkaa siitä kohdepisteeseen* (X, Y, Z)*. [2]*

Kertomalla yhtälön (2) oikean puolen matriisi ja vektori keskenään saadaan kolme skalaariyhtälöä matriisiyhtälöiden sijaan:

$$x = k[m_{11}(X - X_0) + m_{12}(Y - Y_0) + m_{13}(Z - Z_0)]$$

$$y = k[m_{21}(X - X_0) + m_{22}(Y - Y_0) + m_{23}(Z - Z_0)] \quad (3)$$

$$-f = k[m_{31}(X - X_0) + m_{32}(Y - Y_0) + m_{33}(Z - Z_0)]$$

Skaalauskerroin eliminoidaan jakamalla yhtälöryhmän (2) kaksi ensimmäistä yhtälöä kolmannella yhtälöllä, jolloin saadaan kollineaarisuusyhtälöiden esitysmuoto:

$$x = -f\frac{m_{11}(X-X_0)+m_{12}(Y-Y_0)+m_{13}(Z-Z_0)}{m_{31}(X-X_0)+m_{32}(Y-Y_0)+m_{33}(Z-Z_0)} \quad (4)$$

$$y = -f\frac{m_{21}(X-X_0)+m_{22}(Y-Y_0)+m_{23}(Z-Z_0)}{m_{31}(X-X_0)+m_{32}(Y-Y_0)+m_{33}(Z-Z_0)} \quad (5)$$

Yhtälössä (1) voimme viedä orientaatiomatriisin toiselle puolelle ja suorittaa samantyyppiset vaiheet skaalauskertoimen eliminoimiseksi:

$$X - X_0 = (Z - Z_0)\frac{m_{11}(X-X_0)+m_{12}(Y-Y_0)+m_{13}(-f)}{m_{31}(X-X_0)+m_{32}(Y-Y_0)+m_{33}(-f)} \quad (6)$$

$$Y - Y_0 = (Z - Z_0)\frac{m_{21}(X-X_0)+m_{22}(Y-Y_0)+m_{23}(-f)}{m_{31}(X-X_0)+m_{32}(Y-Y_0)+m_{33}(-f)} \quad (7)$$

Näitä yhtälöitä voidaan käyttää tässä muodossa yksinkertaisiin sovelluksiin. Yhtälöitä (4) ja (5) voidaan käyttää kuvakoordinaattien laskemiseen kun ulkoinen ja sisäinen orientaatio sekä kohdepisteen koordinaatit tunnetaan. Yhtälöitä (6) ja (7) voidaan käyttää kun orientaatiomatriisin lisäksi tiedetään kuvakoordinaatti sekä yksi kolmesta kohdepisteen koordinaateista. Näin saadaan ratkaistua kaksi puuttuvaa kohdepisteen koordinaattia.

2.3 Eteenpäinleikkaus ja taaksepäinleikkaus yleisesti

Termillä eteenpäinleikkaus tarkoitetaan kuvassa näkyvän pisteen 3-D kohdekoordinaattien määrittämistä. Tarkoituksena on kohteen rekonstruoiminen eli uudelleenkokoaminen kahden tai useamman sädekimpun avulla. Eteenpäinleikkaus tehdään orientoiduilta kuvilta. Orientointina käy joko kuvaparin keskinäinen orientointi mallikoordinaatistossa tai kummankin kuvan ulkoinen orientointi kohdekoordinaatistossa [4]. Eteenpäinleikkauksessa määritetään kohdeavaruuden kolme kohdekoordinaattia kahden kuvan yhteensä neljästä havaintopisteestä. Jokainen vastinpistepari tuottaa neljä yhtälöä, joista yksi on ylimääräinen. Jokainen seuraava lisäkuva tuo mukanaan aina kaksi uutta ylimääräistä yhtälöä

Taaksepäinleikkauksella tarkoitetaan prosessia, jossa kuvan sijainti- ja orientaatioparametrit määritetään suhteessa kohdekoordinaatistoon. Normaalitapauksessa taaksepäinleikkauksen parametrit sisältävät havaintokohteen 3-D koordinaatit ja kolme rotaatiokulmaakulmaa ω, ϕ, κ, jotka kuvaavat kohde koordinaatiston orientaation suhteessa kuvan koordinaatistoon [5]. Taaksepäinleikkausta käytetään tässä työssä laskemaan tietokone mallin koordinaatistossa sijaitsevien piirre- ja hitsaussaumapisteiden kuvakoordinaatit kaikille kameroille.

Vähintään kolme kohdepistettä vaaditaan taaksepäinleikkauksen ratkaisun laskemiseksi yhdelle kuvalle olettaen että kuvan sisäinen orientaatio tiedetään. Pisteet eivät saa sijaita samassa linjassa eli niiden tulee olla epäkollineaarisia.

3. TYÖN LÄHTÖKOHDAT

Työssä kehitettävän konenäköjärjestelmän ohjelmiston toiminta jakautuu kahteen pääfunktioon: laitteiston ohjaamiseen ja datan käsittelyyn. Datan käsittelyn osuutta pyrittiin kehittämään luomalla automaattinen paikan määritys ohjelman käyttämille hitsaussaumoille.

Hitsaussaumat määritettiin aikaisemmin kuvista manuaalisesti käyttämällä ohjelman omaa hitsaussaumatyökalua. Jokainen sauman piste tuli määrittää vähintään kolmelta eri kuvalta käyttäen apuna ohjelman piirtämiä epipolaarivektoreita. Manuaalisen määrittämisen sijasta haluttiin yksinkertaistaa saumojen määrittämistä luomalla hitsaussaumat automaattisesti kuviin. Tähän oli tarkoitus käyttää manuaalisen syötteen sijasta tiedostosta luettavia 3-D koordinaatteja. Ohjelman kehitystehtävän päämääränä oli luoda algoritmi, joka pystyisi määrittämään hitsaussaumat mahdollisimman luotettavasti ja tarkasti.

3.1 Konenäköjärjestelmä

Tässä työssä ohjelmistoa käyttävä konenäköjärjestelmä on kaupallinen Mapvision Qualitygate [6]. Järjestelmä tekee kappaleen dimensioiden mittaukset käyttämällä useamman kameran muodostamaa kamerapilveä. Kamerapilvellä tarkoitetaan usean eri kameran muodostamaa kokonaisuutta, jossa kuvattava kohde ympäröidään kameroilla niin että sen halutut piirteet saadaan kuvattua. Kaikki kamerat ovat saman ohjausyksikön alaisuudessa ja niitä käsitellään yksittäisenä soluna.

Ennen mittausta kamerapilvi kalibroidaan ja sille opetetaan mitattavan kappaleen piirteet. Jokainen kamera kalibroidaan erikseen kalibrointilevyn avulla. Kun piirteiden sijainnit pystytään määrittämään kameroiden kohdeavaruudessa, voidaan kuvadataa transformoida ohjelmiston käyttämään koordinaatistoon. Vertailemalla koordinaatteja pystytään toteamaan kappaleen oikea muoto.

Kamerapilven lisäksi järjestelmällä on oma kontrolloitu valaistusjärjsetelmä, joka on yhteisessä suljetussa tilassa kamerapilven kanssa tasaisen kuvanlaadun takaamiseksi. Valonlähteiden ja kameroiden sijoittelut suunnitellaan kappaleelta talletettavien piirteiden mukaan.

Qualitygaten suurin etu tavanomaisiin menetelmiin verrattuna on 100% näytteistysotos. Tällä otoksella voidaan havaita erikoisia tai sattumanvaraisia virheitä, joita ei tavanomaisin laadunhallintamenetelmin voitaisi havaita. Yhden kappaleen tarkastamisessa kuluu aikaa noin 10 – 30 sekuntia.

3.2 Ohjelmisto

Hitsaussauman paikan määrittämisessä käytetään Mapvision Toolbox ohjelmaa. Työssä käytetyt hitsaussaumat määritetään ohjelmassa koordinaattipistejonoina. Pistejonojen käyttö on käytännöllistä sekä ohjelmoinnin, että sauman jaottelun kannalta.

Ohjelman varsinainen tehtävä on ottaa kappaleen 3-D mallista valitut virtuaalipiirteet ja kohdistaa ne kuvadatasta löytyviin todellisiin piirteisiin. Tämä tehdään transformoimalla valittuja kiinniottopisteitä mitta-avaruudesta ohjelman kohdeavaruuteen. Tähän vaaditaan eteenpäinleikkauksen mukaisesti vähintään kolme epäkollineaarista kiinniottopistettä. Kiinniottopiste on yleensä transformoinnissa käytettävä reikä tai kulmapiirre.

Ohjelma tarvitsee aina käynnistymisensä jälkeen mittaustilanteen 3-D datan. 3-D datalla tarkoitetaan tässä kohteen tietokonemallia, virtuaalikameradataa, kiinniottopisteiden dataa sekä hitsaussaumojen pisteitä. Käytetyn 3-D datan oletetaan olevan valmiiksi muutettu ohjelman käyttämään koordinaatistoon.

Kamerapilveltä saatava kuvadata luetaan ohjelmaan erillisenä tiedostona. Tiedosto sisältää kaksi kuvaa jokaiselta kameralta, koska kamerapilven käyttämällä valaisutuksella on kaksi eri asetusta etu- ja takavalaistukselle. Näistä kahdesta kuvasta käytetään sitä joka tuo tarkasteltavan piirteen paremmin esiin.

4.2 Muut työkalut

Ohjelmiston kehityksessä käytettiin IDE:nä (Integrated Development Environment) eli kehitysympäristönä Microsoft Visual Studio 2012 Professional ohjelmaa. Ohjelman muutokset ja testaus toteutettiin tässä ympäristössä.

Mapvision Toolbox käyttää 3-D mallien muokkausohjelmana Siemens NX ohjelmistoa. Ohjelmalla voidaan valita käsiteltävät kiinniottopisteet ja hitsaussaumat XML tiedostomuotoon, jonka jälkeen ne muutetaan Mapvision Toolbox avaruuteen. Kun pisteet on muutettu ne voidaan ladata Mapvision Toolbox ohjelmaan ja ottaa suoraan käyttöön.

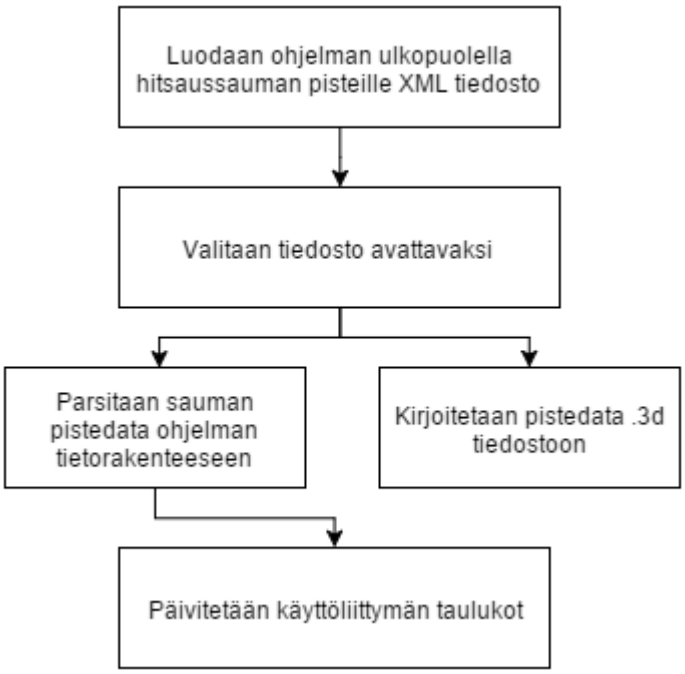
4. TYÖN SUUNNITTELU

Ennen toteutusta tehtiin vaatimusmäärittely uudelle toiminnallisuudelle. Kaikkiin työn puitteissa esiin tulleisiin kehitystarpeisiin ei työn laajuudessa pystytty vastaamaan, mutta esille tulleet kehitystarpeet dokumentoitiin jatkokehitystä varten. Lähdekoodiin tehdyille muutoksille ja lisäyksille kirjoitettiin englanninkielinen kommentointi.

4.1 Vaatimusmäärittely

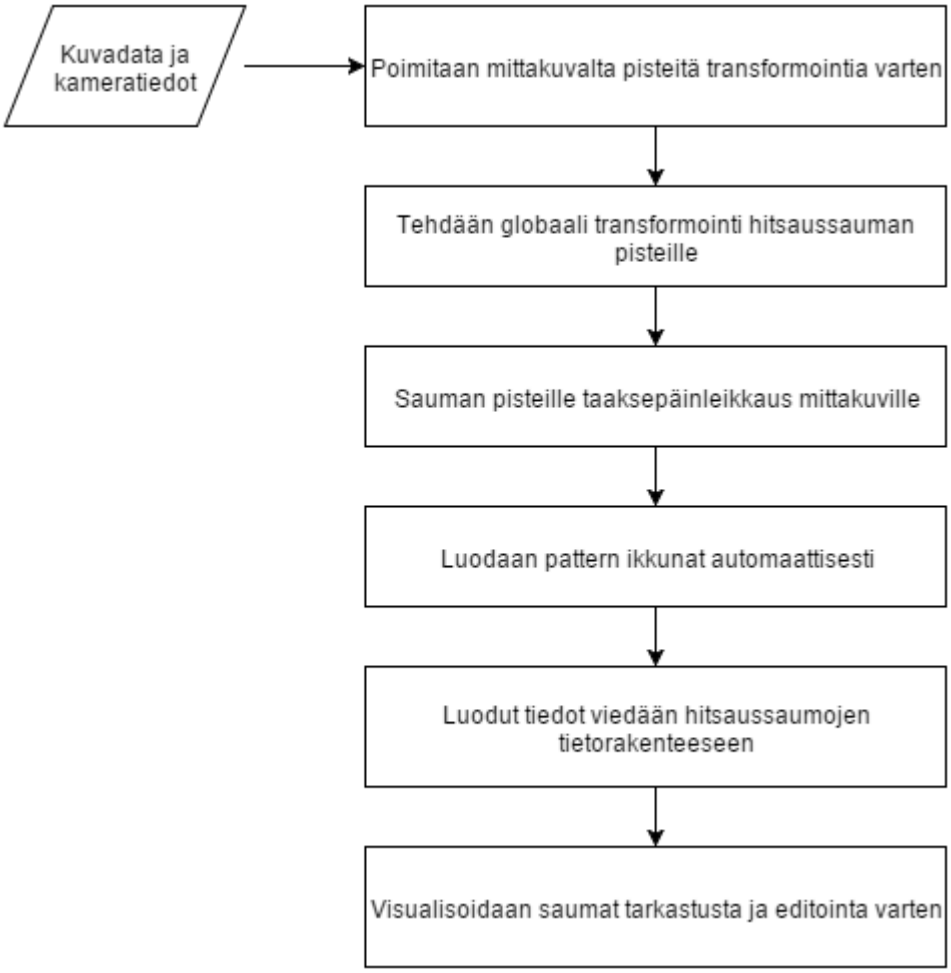
Uuden toiminnallisuuden selkeyttämiseksi ohjelmalle tehtiin kaksivaiheinen vaatimusmäärittely. Lisättävä toiminnallisuus haluttiin jakaa mahdollisimman yksinkertaistettuihin osiin, joista jokaisen tuli vastata yhtä ohjelmointitehtävää. Tällöin työn laajuus saatiin rajattua järkeväksi. Ohjelman monimutkaisuuden lisäämisen välttämiseksi pyrittiin käyttämään mahdollisimman paljon käytössä olevia ohjelman rakenteita.

Ensimmäisenä vaiheena ohjelmiston muokkaamisessa oli tietorakenteen luominen hitsaussaumoille sekä tiedon lukeminen tähän tietorakenteeseen. Kun hitsaussauman pistedata saatiin luettua ohjelman uuteen tietorakenteeseen voitaisiin sitä käyttää valmiiden geometristen laskentafunktioiden alustukseen. Tiedon lukeminen ohjelmaan toteutettiin kuvan 1 esittämän diagrammin mukaisesti.



Kuva 1. *Ohjelmointivaatimukset hitsaussauman lukemiseksi ohjelman tietorakenteeseen.*

Toisena osuutena ohjelmoinnissa oli luetun datan transformointi oikeaan koordinaatistoon ja tulosten visualisointi. Tätä varten ohjelmaan lisättiin uusi työkalu, jonka tuli transformoida luetut hitsaussauman pisteet, sekä luoda niistä segmentit. Uuden työkalun toiminta toteutettiin kuvan 2 esittämän diagrammin mukaisesti.

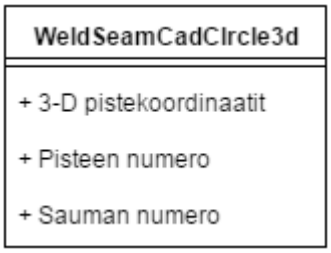


Kuva 2. *Ohjelmointivaatimukset sauman segmenttien luomiseksi ja visualisoimiseksi uudella työkalulla. Kameroiden tiedot ja kuvadata saadaan ladattua erillisistä tiedostoista.*

4.2 Hitsaussauman tietotyyppi

Hitsaussaumalle lisättiin uusi luokka, jota käytettiin sauman pisteiden käsittelemiseen ohjelman tietorakenteissa. Uuden luokan lisääminen ohjelmistoon vaati huomattavan määrän muokkaamista, jotta olemassa olevat menetelmät olivat yhteensopivia sen kanssa. Uusi luokka periytettiin reikäpiirteille valmiiksi käytössä olevasta luokasta. Luokkaa käytettiin sekä kiinniottopisteiden että hitsaussaumojen pisteiden käsittelyyn. Luokan olio pitää sisällään hitsaussauman yhden pisteen tiedot. Jatkokehityksen kannalta luokkaa

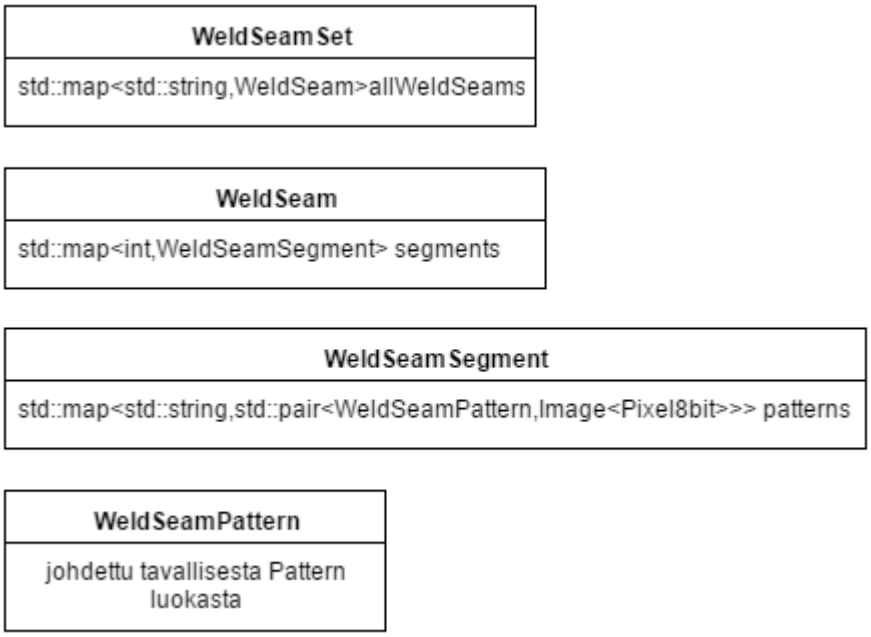
voitaisiin muokata pitämään sisällään tämän lisäksi muuta hitsaussauman tietoa kuten esimerkiksi sauman pintamuotoa kuvaavia parametreja.



Kuva 3. *Hitsaussauman pisteille lisätty luokka.*

Hitsaussauman jokainen piste talletetaan luettaessa *map* tyyppisen tietorakenteen elementtiin. Yksi elementti pitää sisällään avain-arvo parin. Avaimena toimii pisteen nimi ja arvona uusi luokkatyyppi.

Kun hitsaussauman pisteet on luettu ohjelmaan siirrytään transformoimaan pisteitä. Transformoinnin jälkeen ohjelma jaottelee hitsaussauman segmentteihin. Hitsaussauman segmentti ymmärretään tässä hitsaussauman kahden peräkkäisen pisteen kattavana sylinterimäisenä 3-D objektina. Yksi segmentti rakentuu suorakulmaisista 2-D *WeldSeamPattern*-olioista, jotka luodaan jokaisen segmentin havaitsevan kameran kuvatasoon. Pattern olio tulee nimetä yksilöllisesti ohjelmassa tapahtuvien päällekkäisyyksien välttämiseksi. Olion nimi-tieto sisältää saumanumeron, segmentin järjestysluvun sekä kameranumeron. Sauman alku- ja loppupistettä lukuun ottamatta jokainen hitsaussauman piste kuuluu kahteen segmenttiin niiden alku- tai loppupisteenä.



Kuva 4. *Ohjelman kaikille hitsaussaumoille tarkoitettu tietorakenne. Yksi segmentti pitää sisällään jokaiselle kameralle oman 2-D WeldSeamPattern olion.*

5. TYÖN VAIHEET

5.1 Datan lukeminen ohjelmaan

Hitsaussaumojen pistedata saadaan luomalla saumat Siemens NX ohjelmistolla kappaleen 3-D malliin. Siemens NX:ssä valittujen piirteiden sisältää hitsaussaumojen lisäksi myös vähintään kolme kiinniottopistettä globaalia transformointia varten. Valitut piirteet talletetaan XML tiedostoon. Oletetaan että Siemens NX:ssä valitut pisteet transformoidaan Mapvision Toolbox koordinaatistoon.

Luotu XML tiedosto määrittää kiinniottoreiät niiden kehän pisteiden perusteella. Ohjelma tekee myöhemmin näkyvyysanalyysin perustuen reiän pintanormaalin tasolla olevaan keskipisteeseen. Hitsaussauma määritetään XML tiedostoon 3-D koordinaattien pistejonona. Näkyvyysanalyysiä varten jokaiselle reikäpiirteelle lasketaan keskipiste $C_o(x_o, y_o, z_o)$, joka saadaan laskemalla keskiarvo reiän kehän pisteiden avulla:

$$x_o = \frac{\sum_{i=1}^{n} X}{n}$$

$$y_o = \frac{\sum_{i=1}^{n} Y}{n} \qquad (7)$$

$$z_o = \frac{\sum_{i=1}^{n} Z}{n}$$

missä X, Y ja Z ovat reiän kehän pisteitä, x_o, y_o ja z_o ovat keskipisteen koordinaatit ja n on kehän pisteiden määrä.

Ohjelmassa käytetään TinyXml nimistä avoimen lähdekoodin tekstiparseria [7], joka jäsentelee XML tiedoston sisältämän tiedon luettavaksi ja muokattavaksi ohjelman kielellä.

5.2 Piirteiden määrittäminen

Saumadata luetaan sisään ohjelman kameratyökalussa. Luettu saumadata esitetään automaattisesti ohjelman työkalun 3-D näkymässä yhdessä virtuaalikameroiden ja kappaleen 3-D mallin kanssa. Kameratyökalu on ohjelman virtuaalityökalu, jossa ladataan segmenttien luomiseen vaadittava 3-D data. Ladatun saumadatan sisältämiä piirrepisteitä voidaan tässä vaiheessa ottaa käyttöön tai poistaa käytöstä. Piirteet esitetään 3-D näkymässä kappaleen tietokonemallin kanssa. Yksi piirre näkyy sinisenä pallo-objektina, kun taas hitsaussauma esitetään pallojonona. Yhden pallon säde on suoraan verrannollinen XML tiedoston määrittelemän piirrepisteen säteeseen. Hitsaussauman pisteitä esittävien objektien säde on vakioitu ohjelmassa kokonaislukuun kaksi.

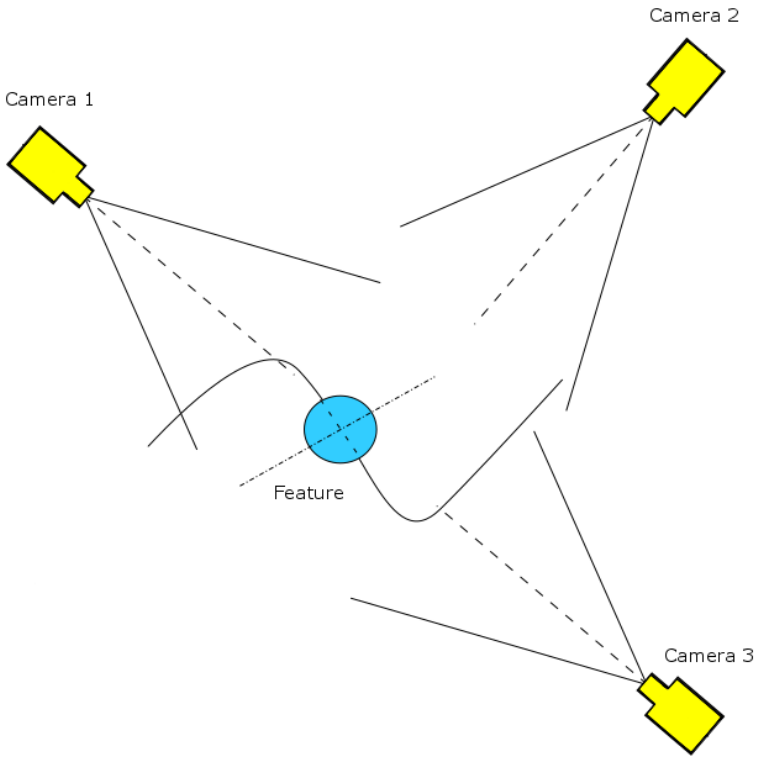
Työkalulla luodaan virtuaalinäkymä kappaleen mittaustilanteesta. Tällä tavalla saadaan vertailukelpoinen normaali, johon mittakuvalta saatavaa dataa lopuksi verrataan.

5.3 Näkyvyysanalyysi

Näkyvyysanalyysillä tarkoitetaan ohjelmiston tekemää näkyvyystestiä valituille piirteille kunkin virtuaalikameran kuvakulmasta. Analyysi käy läpi kunkin virtuaalikameran näkymän ja merkitsee kaikki tämän näkemät piirteet muistiin, mikäli piirteen osoittama 3-D objekti näkyy virtuaalikameralle. Näkyvyysanalyysi on nopea ja yksinkertainen tapa luokitella piirteille käytettävät kamerat kamerapilvestä. Yhden piirteen näkyvyysanalyysi luokitellaan sitä paremmaksi mitä useampi virtuaalikamera sen havaitsee.

Yksinkertaisuus aiheuttaa positiivisia virheluokituksia kun 3-D objekti näkyy osittain virtuaalikameralle, mutta piirre ei. Nämä tilanteet luokitellaan näkyvyysanalyysissä posiitivisiksi, mutta aiheuttavat turhaa laskentaa segmenttien luomisvaiheessa. Aiheutuva virhe voidaan havaita vasta saumojen visualisoinnin jälkeen. Virhe tapahtuu kun virtuaalikameran havaintovektori piirteelle on melkein kohtisuorassa kulmassa piirteen pintanormaalin suhteen. Virhetilanne on havainnollistettu kuvassa 5.

Näkyvyysanalyysin yhteydessä ohjelma tallettaa kaikkien virtuaalikameroiden kuvanäkymät ohjelman tietorakenteeseen. Yhdellä virtuaalikameralla talletetaan kaksi näkymää, joista toiseen kuvaan luodaan keinotekoinen linssivääristymä. Näitä kahta kuvaa käytetään näkyvyysanalyysin jälkeen luomaan kamera-piirre parit sillä ehdolla, että 3-D objekti on näkyvissä kummassakin kuvassa. Pareja käytetään myös segmenttien luomisvaiheessa kahden peräkkäisen hitsaussauman pisteen löytämiseen.

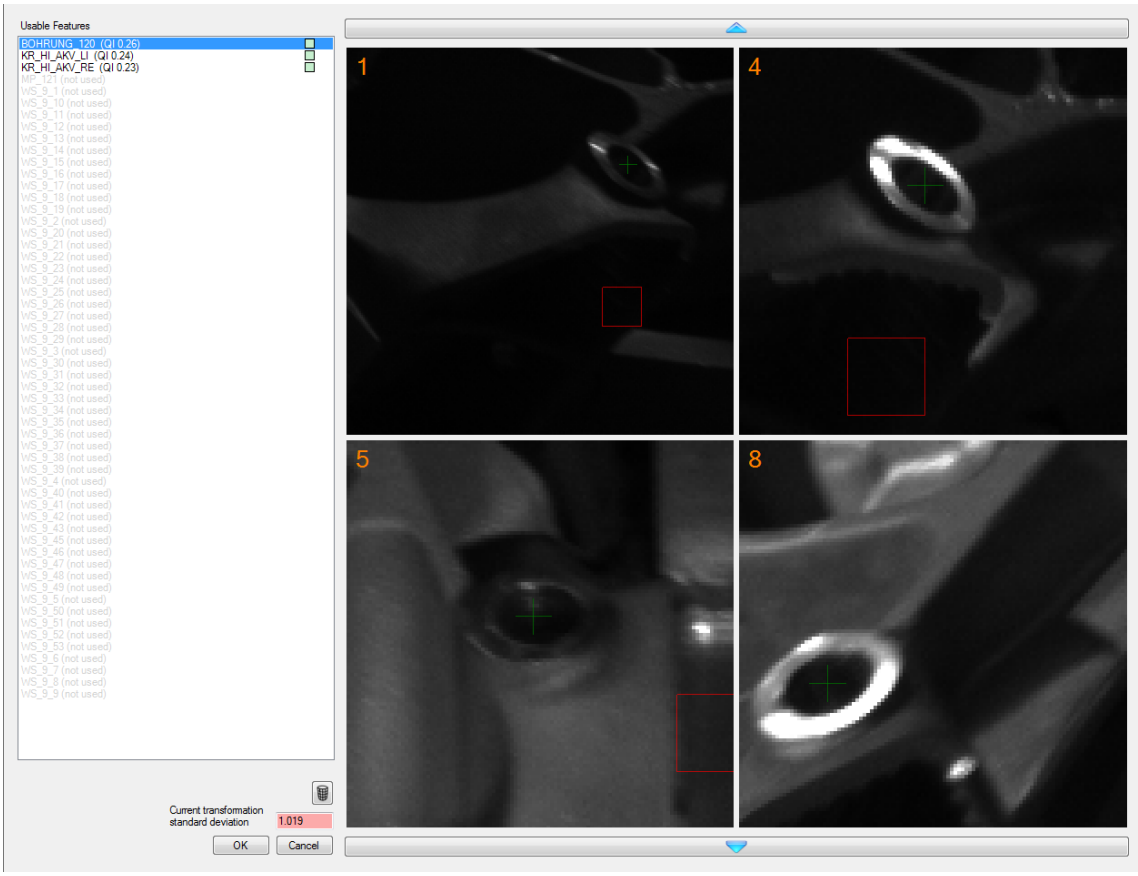


Kuva 5. *Poikkileikkaus näkyvyysanalyysin ongelmatilanteesta. Sininen objekti on reikäpiirteen osoittava pallo, joka näkyy kameroille 1 ja 2. Todellinen piirre näkyy vain kameralle 2. Kamera 3 ei näe objektia eikä piirrettä.*

5.4 Pisteiden valitseminen transformointia varten

Näkyvyysanalyysin jälkeen valitaan piirrepisteiden sijainti kuvadatalta käyttäen uutta työkalua. Työkaluun ladataan erillinen mittakuvatiedosto piirrepisteiden valitsemista varten. Pisteiden globaaliin transformointiin käytetään reikäpiirteitä. Valittujen reikäpiirteiden sijainnit mittakuville lasketaan käyttäen kollineaarisuusehtoa ja taaksepäinleikkausta.

Taaksepäinleikattujen pisteiden sijainnit esitetään uuden työkalun näkymässä, jossa on kameranäkymien lisäksi lista kaikista XML tiedostosta luetuista piirrepisteistä (Kuva 6). Työkalu esittää listasta valitun piirrepisteen sijainnin taaksepäin leikattuna kameran mittakuville sekä sille luodut pattern ikkunat. Piirteen todellinen sijainti valitaan mittakuvalta käyttäjän toimesta. Valitun sijainnin kuvakoordinaatti piirtyy vihreänä rastina kuvanäkymään.



Kuva 6. *Kuvakaappaus Testing Tool työkalun näkymästä. Työvaiheena reikäpiirteiden määrittäminen mittakuvilta.*

Eteenpäinleikkauksen tekemiseksi kolme kiinniottopistettä tulee määrittää kolmen eri kameran näkymästä. Ohjelma laskee transformaatioparametrit vanhan pisteen ja uuden pisteen välillä siirtymiseksi aina kun yksi reikäpiirre määritetään mittakuvalta. Kun kolme eri reikäpiirrettä on määritetty vähintään kolmelle eri kameralle voi käyttäjä hyväksyä uudet transformaatioparametrit. Hyväksynnän jälkeen ohjelma tallettaa parametrit erilliseen *TransformationParameters* nimiseen tekstitiedostoon. Tekstitiedostosta luettavia parametreja käytetään transformoimaan sisäänluetut hitsaussauma- ja piirrepistekoordinaatit todellisuutta vastaavaan koordinaatistoon. Näin saadaan uusi kohdekoordinaatti jokaisen piirteen ja hitsausssauman pisteelle.

5.5 Segmenttien luominen

Pisteille tehdyn globaalin transformoinnin jälkeen taaksepäinleikataan uusia kohdekoordinaatteja mittakuville. Taaksepäinleikkaus suoritetaan pisteelle jos näkyvyysanalyysissä on luotu kamera-piirre pari. Kahden peräkkäisen kohdepisteen taaksepäinleikkauksella lasketut pisteet tallennetaan *WeldSeamPattern*-olioihin.

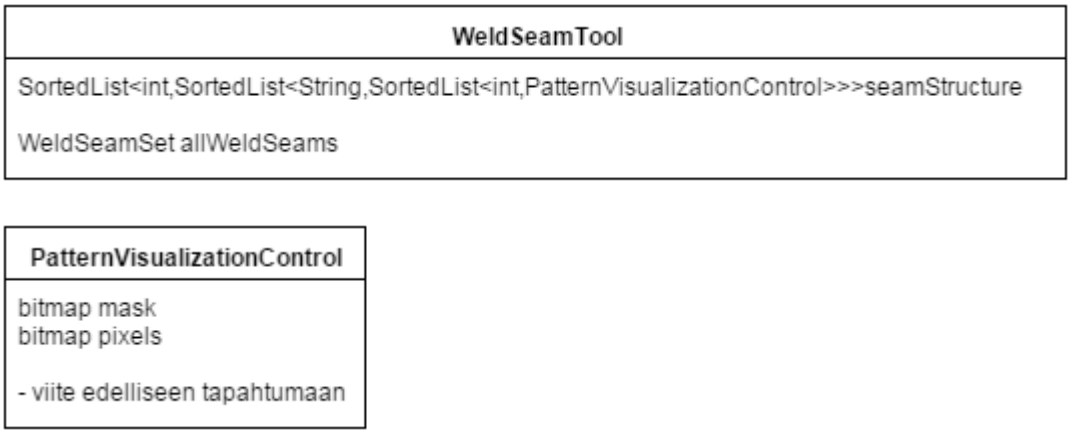
WeldSeamPattern-olio sisältää muun muassa alku- ja loppupisteen kuvakoordinaatteina. Yksi olio luodaan kahdelle peräkkäiselle sauman pisteelle sillä ehdolla että kumpikin

pisteistä nähtävissä samalla kameralla. Mikäli tämä ehto ei täyty, aloittaa ohjelma tarkistuksen seuraavasta hitsaussauman pisteestä. Olioiden luominen suoritetaan jokaiselle kameranäkymälle. Näitä 2-D olioita käyttämällä alustetaan hitsaussaumojen aikaisemmin määritelty tietorakenne, missä yksi *WeldSeamPattern*-olio edustaa kahden sauman pisteen välille luotua segmenttiä 2-D tasossa. Kahden pisteen välille luotu segmentti olio koostuu jokaisen kameran näkymään luodusta pattern-oliosta.

5.6 Segmenttien visualisointi

Kun hitsaussaumojen tietorakenne on saatu alustettua voidaan oliot esittää ohjelman hitsaussaumatyökalussa. Esittämistä varten saumojen tietorakenne luetaan ohjelman piirtotyökalun käyttämään listarakenteeseen.

Listarakenne käyttää PatternVisualizationControl tyyppisiä tapahtumia WeldSeamPattern olidoiden piirtämiseksi. Yksi tapahtuma luodaan lukemalla sauman kaksi peräkkäistä pistettä hitsaussasuman tietorakenteesta listarakenteeseen.



Kuva 7. *Hitsaussaumojen visualisoinnissa tarvittava listarakenne*

Segmentit visualisoidaan ohjelman hitsaussaumatyökalun näkymään, missä niitä voidaan muokata ja tarkastella eri kameroiden näkymistä. Aktiiviseksi merkatuissa kameranäkymissä kuvassa on nähtävissä ohjelman luomat hitsisauman pattern ikkunat. Ohjelman tuottaman visualisoinnin hyväksyminen jätettiin loppukäyttäjälle.

6. TESTAUS JA TULOKSET

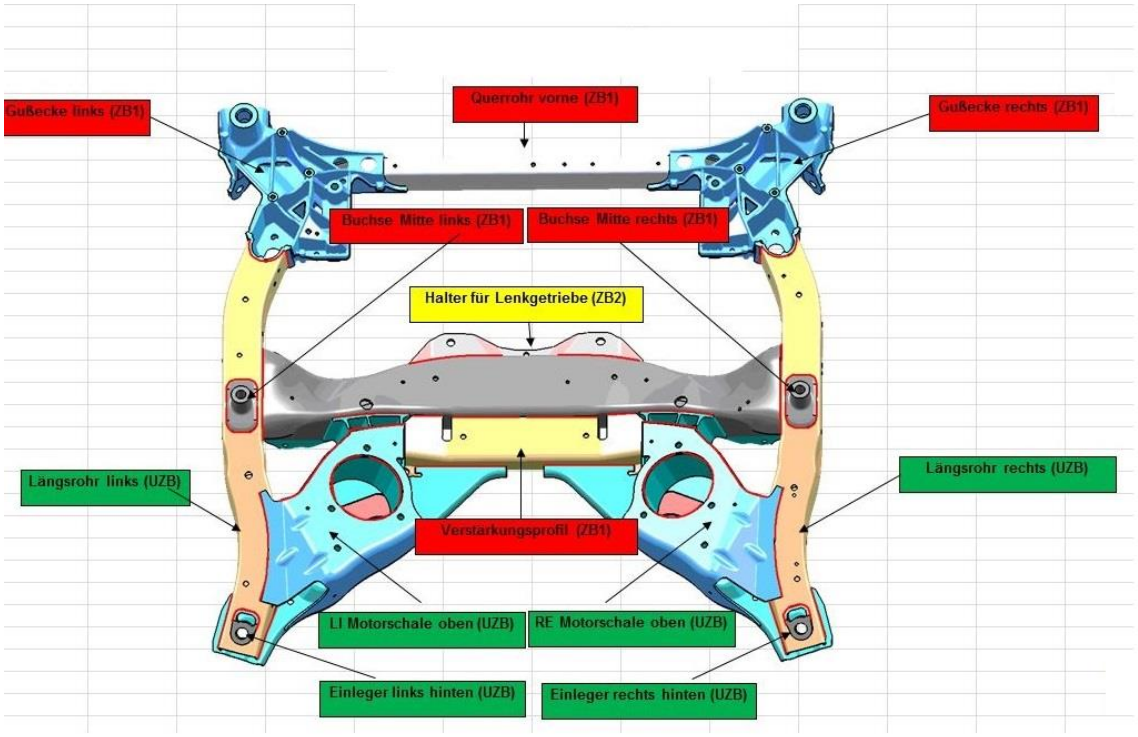
Esitellään ohjelmalla ajettava testidata sekä tarkastellaan globaalisti transformoitujen hitsaussaumojen pisteiden ja niille luotujen segmenttien tuloksia eri kameranäkymissä. Globaalilla transformoinnilla tarkoitetaan kaikille pisteille tehtävää transformaatiota, joka tehdään kolmea globaalia kohdepistettä käyttäen. Kaikille kappaleen saumoille ei pystytty ohjelman rajallisen toiminnallisuuden takia luomaan segmenttejä.

6.1 Testidatan esittely

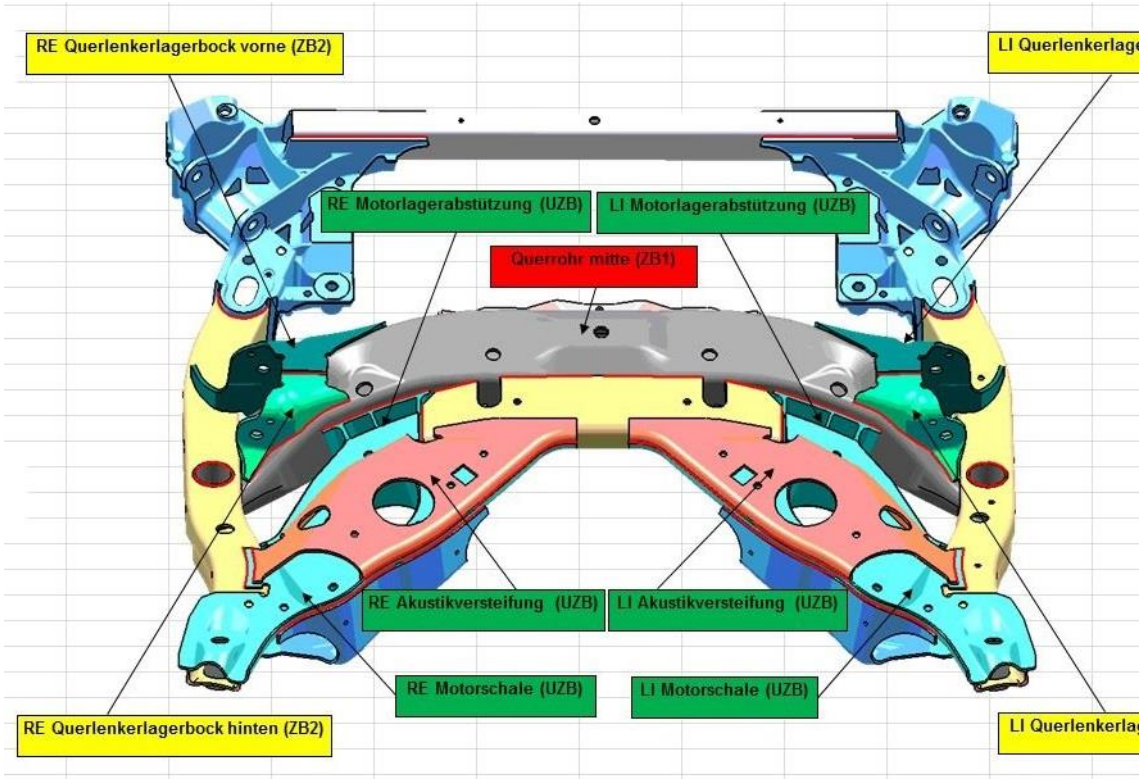
Ohjelman testaamiseksi käytettiin erään autonvalmistajan tuottaman auton osan 3-D mallia sekä sitä vastaavasta sarjatuotantokappaleesta tallennettua kuvadataa. Kuvadata ja sen tallentaneen kamerajärjestelmän kalibrointitiedot saatiin konenäön valmistajan tietokannasta. Käytetty kamerajärjestelmä sisälsi yhteensä 36 kameraa.

Sarjatuotantokappaleen valmistuksessa käytetään kolmea kokoonpanovaihetta. Kokoonpanovaiheet on nimetty tunnisteilla ZB1, ZB2 ja UZB. Kokoonpanovaiheet on merkitty myös eri väritunnistein. Kappaleeseen tulevat hitsaussaumat on merkitty punaisella reunaviivalla.

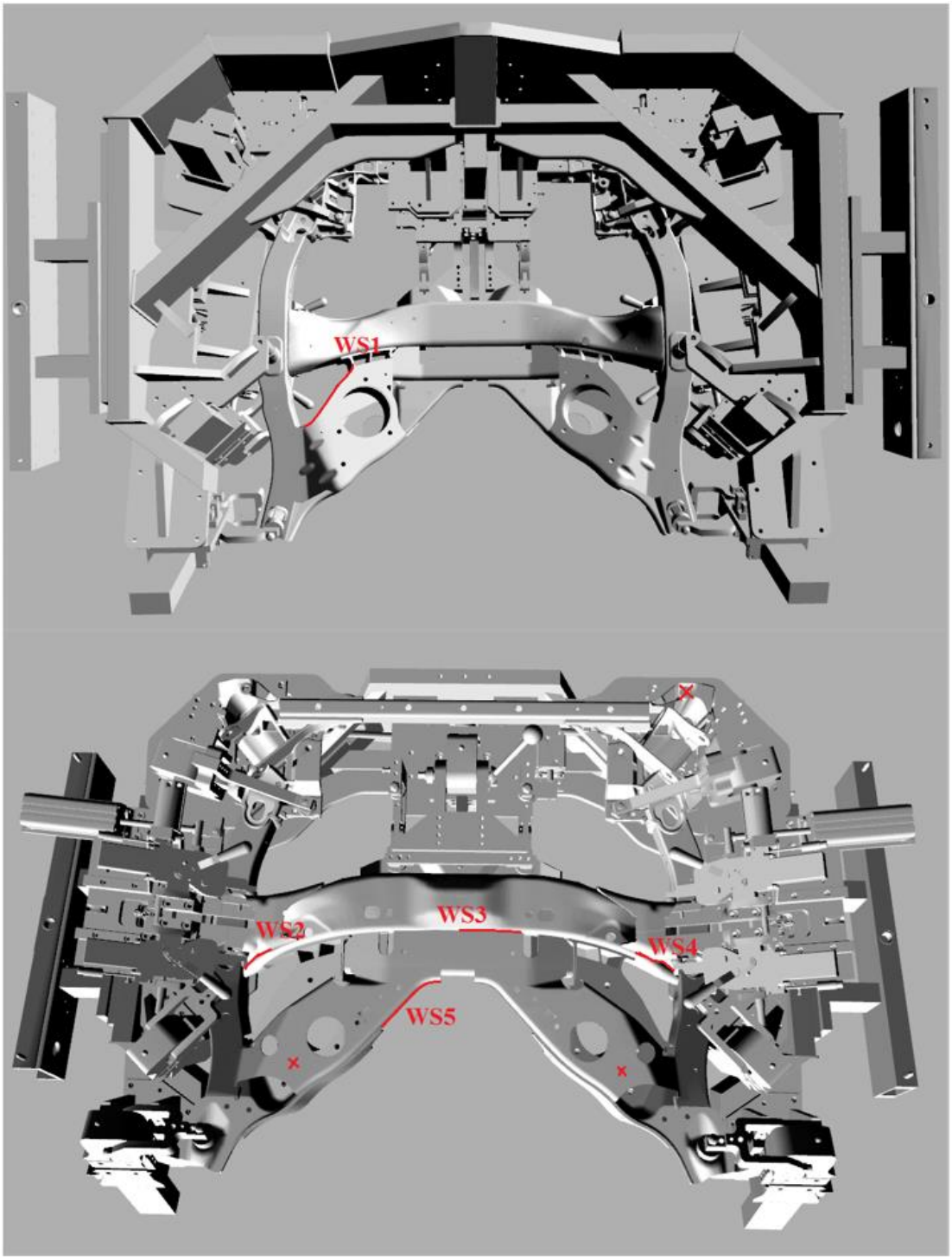
Testattavat hitsaussaumat (Kuva 10) valittiin kattamaan kaikki kolme kokoonpanovaihetta. Ohjelman rajallisen toiminnallisuuden takia kaikkia kappaleen saumoja ei pystytty testaamaan.



Kuva 8. *Sarjatuotantokappaleen osakuva yläpuolelta. Punainen kokoonpanovaihe ZB1, keltainen ZB2 ja vihreä UZB.*



Kuva 9. *Sarjatuotantokappaleen osakuva alapuolelta.*



Kuva 10. *Kappaleesta ohjelman testaukseen valitut hitsaussaumat WS1, WS2, WS3, WS4 ja WS5. Kiinniottopisteiden sijainti on merkitty kuvaan punaisilla rasteilla.*

Globaali transformointi tehdään käyttäen kappaleen reunoilla sijaitsevia kiinniottopisteitä. Mitä useamman liitossauman kautta tarkasteltava hitsaussauma on liitoksissa kiinniottopisteiden määrittämään kehykseen, sitä todennäköisemmin luotujen segmenttien sijainti poikkeaa todellisen sauman sijainnista. Ohjelman avulla pyrittiin luomaan vertailukelpoista informaatiota ideaalisen hitsaussauman ja todellisen hitsaussauman sijainnin välillä. Informaation avulla voidaan tehdä johtopäätöksiä globaalin transformoinnin soveltuvuudesta hitsaussauman paikan määrittämiseksi.

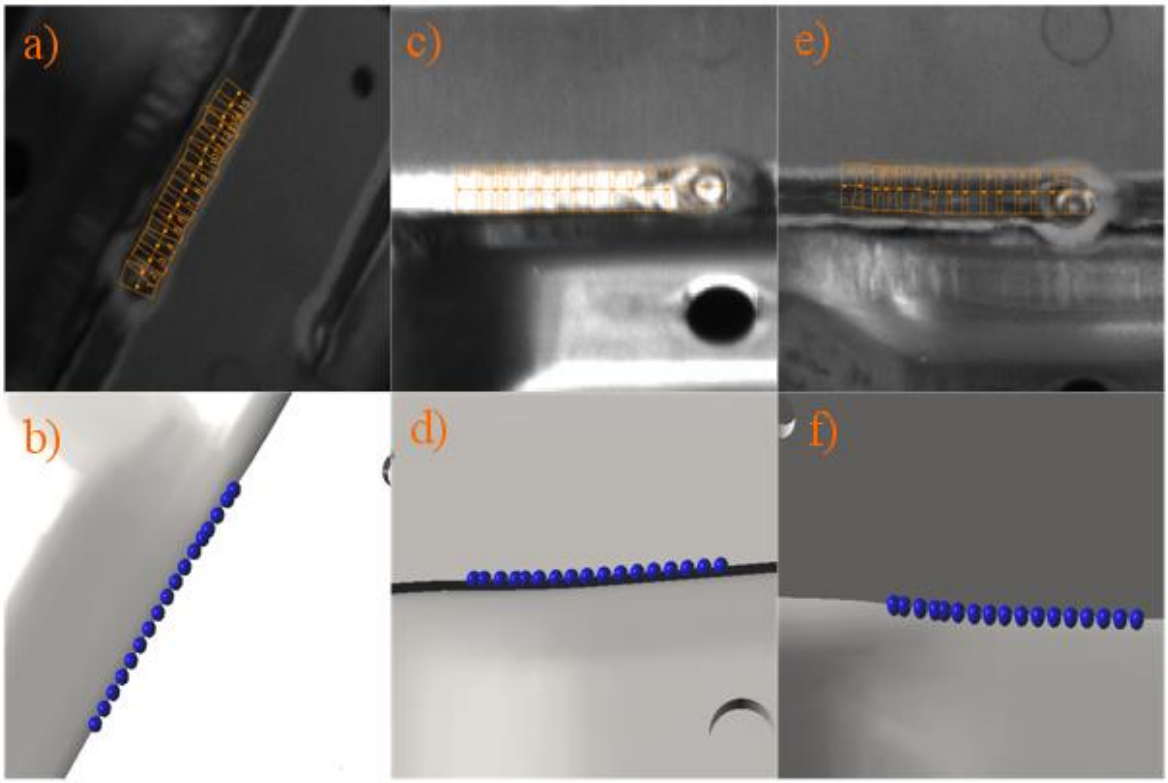
Mittakuvista valittiin ne kameranäkymät joiden katsottiin antavan havainnollistavimmat tulokset ohjelman toimivuudesta. Visualisointityökalun näkymässä esitetään aina yksi hitsaussauma segmentoituna kameralle. Pienet oranssit pisteet ovat ohjelmaan ladatun hitsaussauman pisteet transformoituna kameran kuvatasoon. Kaksi peräkkäistä pistettä esittävät yhden segmentin alku- ja loppupistettä..

6.2 Automaattisen segmentoinnin testaaminen

Fyysiset kappaleet ovat harvoin muodoltaan identtisiä 3-D mallin kanssa. Transformoimalla hitsaussauman pisteitä ensin globaalisti ja sen jälkeen leikkaamalla taaksepäin niitä kuville, luodaan näkymä siitä kuinka yhdenmukainen hitsaussauman sijainti on todellisen sauman kanssa. Hitsaussauman paikan määrittämisen onnistumiseen vaikuttaa transformoinnin lisäksi kappaleen oikea muoto, joka on riippuvainen kokoonpanoprosessista ja lämpölaajenemisesta. Näiden kahden viimeisen vaikutusta ei työn puitteissa pystytty todentamaan. Tulosten tarkastelun kappaleissa käydään läpi mittakuville segmentoitu hitsaussauma ja verrataan sitä virtuaalikameroiden ideaaliseen näkymään.

6.2.1 Ensimmäisen kokoonpanovaiheen sauma

Ensimmäisen kokoonpanovaiheen osat ovat merkitty osakuvaan punaisella ZB1 tunnisteella. Tästä kokoonpanovaiheesta otettiin tarkasteltavaksi hitsaussauma WS3.

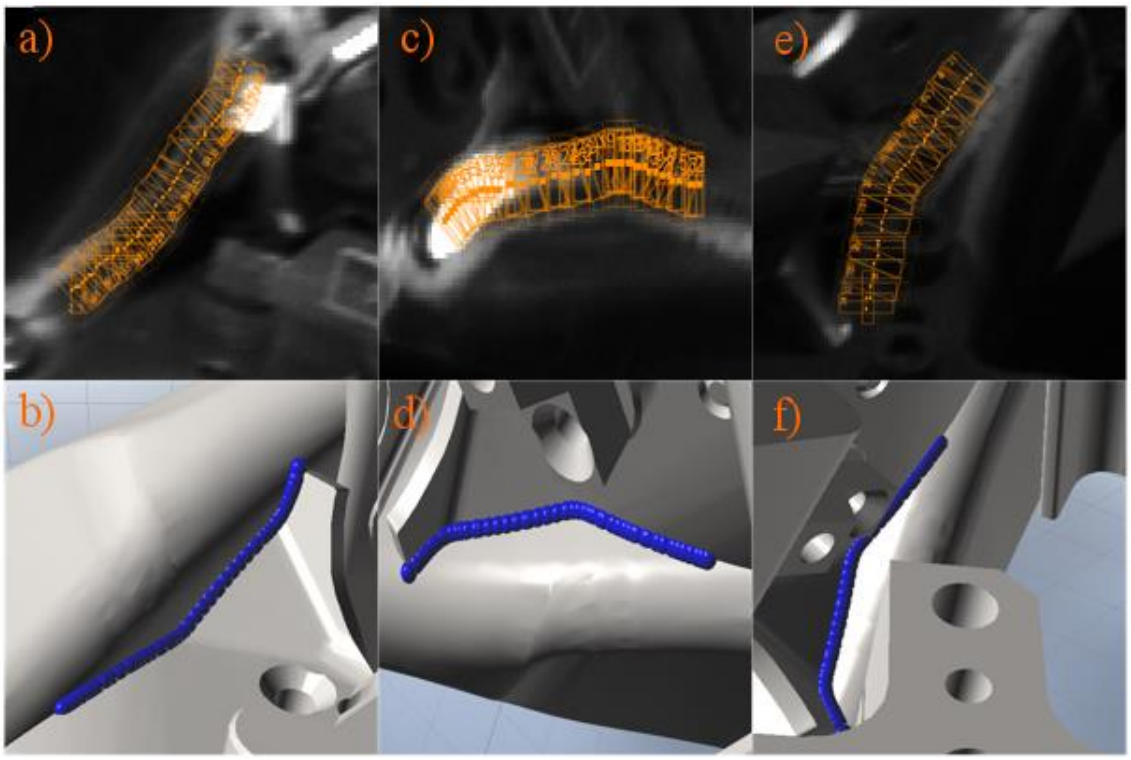


Kuva 11. *Hitsaussauma WS1 segmentoituna kolmen eri kameran mittakuvalle.*

Tarkasteltavan hitsaussauman segmentointi osuu sauman päälle kuvan c) näkymässä. Kuvien 11.a) ja 11.e) sauman ja segmentoinnin tuloksen loppupäässä on havaittavissa poikkeama virtuaalisen ja todellisen sauman paikan välillä. Poikkeama johtuu todellisen hitsausradan lineaarisuudesta verrattuna virtuaalisen sauman muotoon. Tämän sauman tapauksessa globaali transformointi ei ole riittävä, koska halutunlaiseen segmentointiin ei päästä ilman manuaalisia korjauksia.

6.2.2 Toisen kokoonpanovaiheen saumat

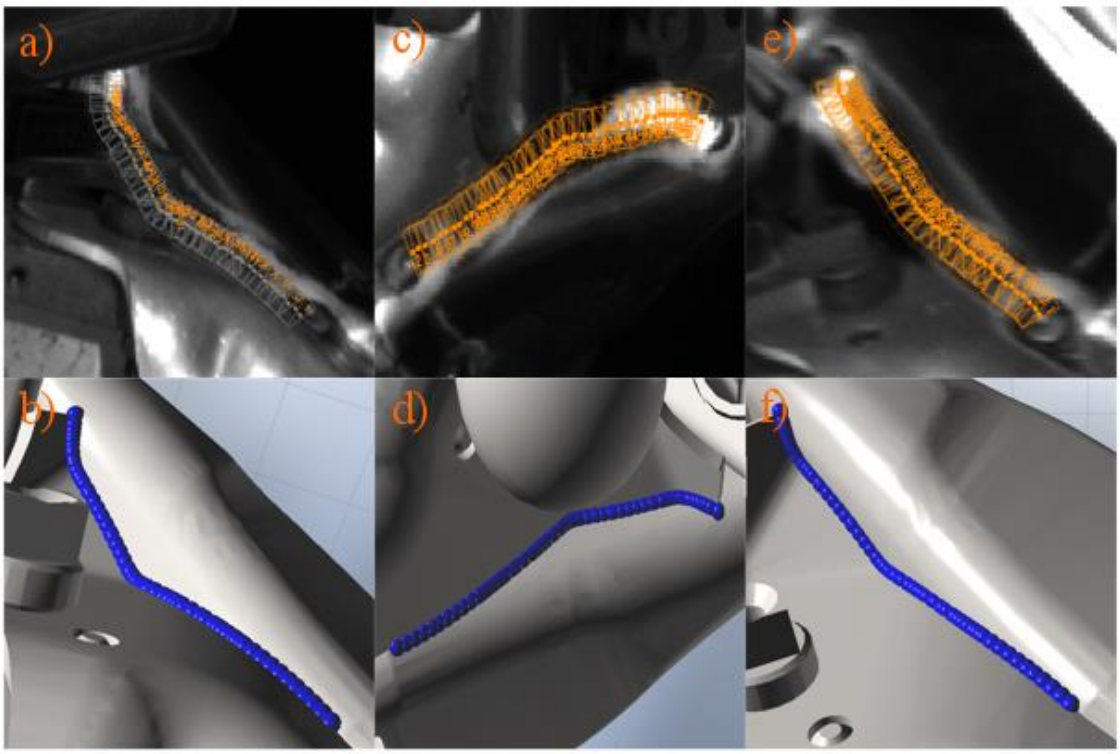
Toisen hitsausvaiheen saumat ovat merkattu osakuvaan keltaisella ZB2 tunnisteella. Kaksi seuraavaa hitsaussaumaa WS2 ja WS4 ovat ideaalisessa tilanteessa symmetrisiä toisiinsa nähden.



Kuva 11. *Hitsaussauman WS2 segmentointi kolmelle eri mittakuvalle.*

Sauman WS2 pisteiden sijainti vastaa sauman keskiosuuden osalta hyvin todellista sauman sijaintia. Segmentoidun sauman alku- ja loppupisteen sijainti puolestaan poikkeavat todellisen sauman pisteistä. Tarkasteltavan sauman transformoinnissa olisi järkevämpää käyttää lokaalimpaa transformointia valittaessa pisteitä transformointia varten. Ohjelman visualisointityökalu ei myöskään pysty esittämään kaikkea informaatiota selkeästi lähekkäin määritettyjen hitsaussauman virtuaalipisteiden tapauksessa. Tämä tekee automaattisen segmentoinnin jälkeisestä manuaalisesta editoimisesta haastavaa.

Kuvassa 11.e) viimeinen pattern ikkuna on kohtisuorassa edelliseen verrattuna, vaikka niiden tulisi olla enemmän yhdensuuntaiset. Virhetilanne ilmenee taaksepäinleikkauksen jälkeen luotaessa WeldSeamPattern olioita. Virhe aiheutuu jos sauman pisteitä esittävien objektien säde on suurempi kuin sauman kahden peräkkäisen pisteen välinen etäisyys.

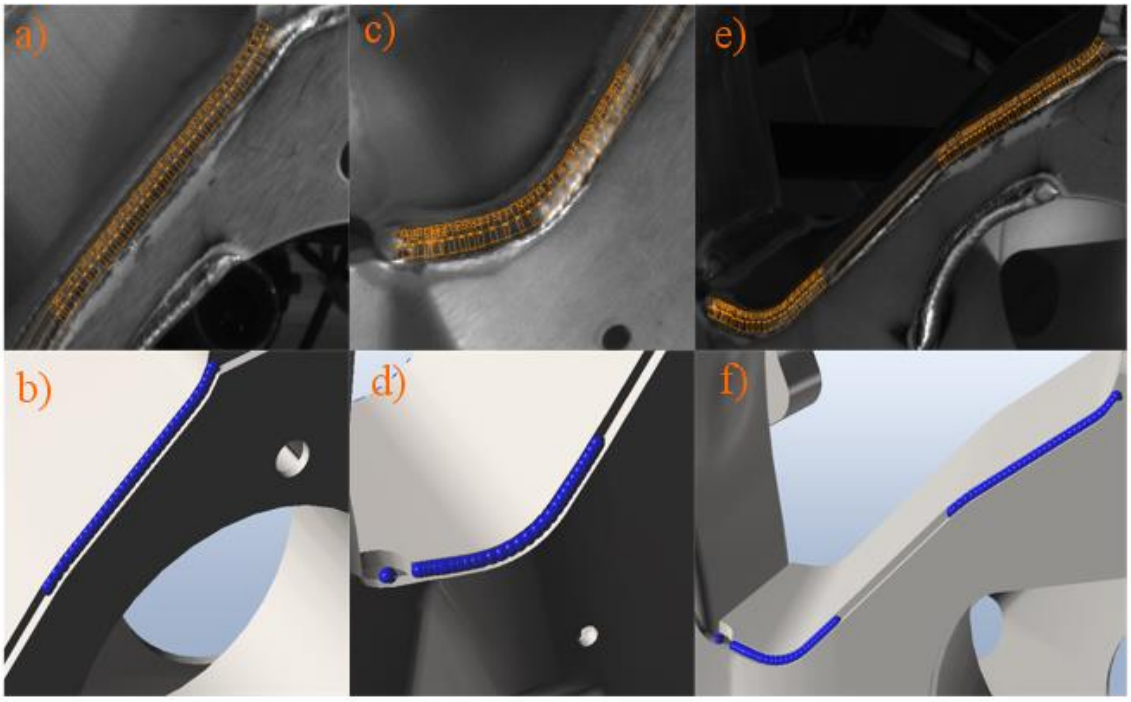


Kuva 12. *Hitsaussauman WS4 segmentointi kolmelle eri mittakuvalle.*

Sauman WS4 segmentoinnin tulos vastaa edellisen sauman tulosta. Virtuaalisauman alku- ja loppupään paikka ei osu kohdilleen. Segmentoinnilla saadut keskivaiheen pisteet osuvat parhaiten kohdilleen kaikissa kolmessa kuvassa. Kuvassa 12.a) segmentoinnin luoma loppupisteen paikka poikkeaa todellisesta saumasta. Segmentoinnin tulos on sauman keskimmäisten pisteiden osalta hyvin lähellä todellista sauman paikkaa.

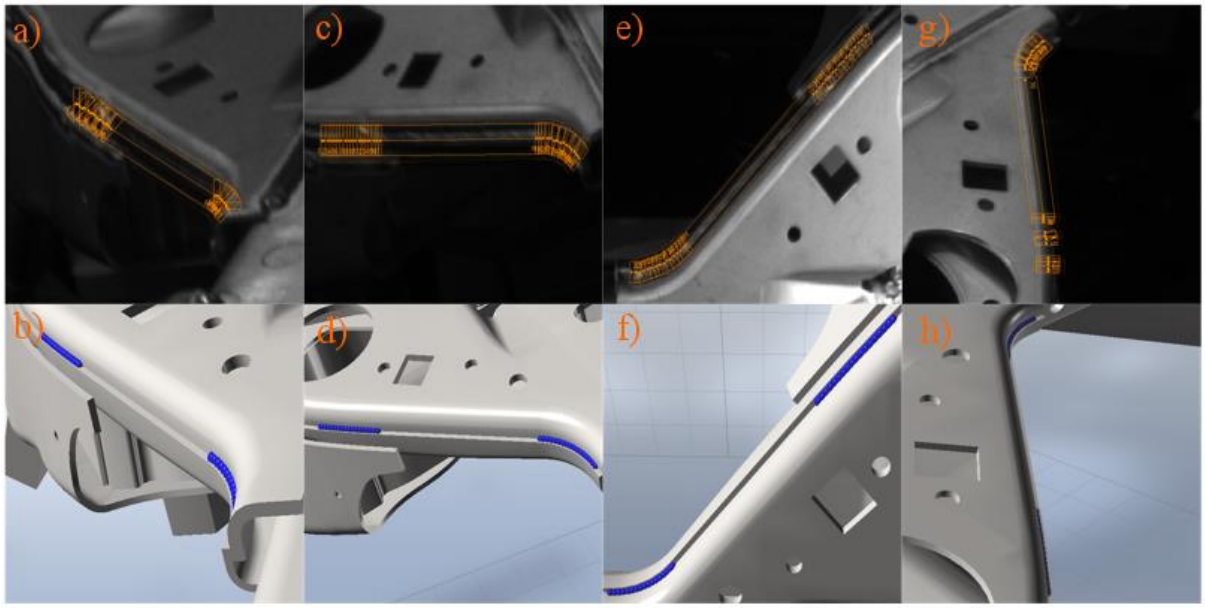
6.2.3 Alikokoonpanon saumat

Alikokoonpanon hitsaussaumat on merkattu osakuvaan vihreällä tunnuksella UZB.



Kuva 13. *Hitsaussauman WS1 segmentointi kolmelle eri mittakuvalle.*

Kuvista 13.c) ja 13.e) huomataan yhden pisteen ja samalla yhden segmentin puuttuminen. Päätepisteitä lukuun ottamatta segmentit osuvat näiden kolmen kameran näkymässä päällekkäin todellisen sauman paikan kanssa. Tarkasteltavan sauman hitsaus on yksiselitteistä kappaleessa olevan railon ansiosta, eikä voi suoralla osuudella vaihdella suuresti. Poikkeama huomataan kuvan 13.a) sauman alkupäässä, jossa todellinen sauma jatkuu vielä viistosti ylös oikealle.



Kuva 14. *Hitsaussauman WS5 segmentointi neljälle eri mittakuvalle.*

Kuvassa 14 näkyvän sauman WS5 segmentoinnin tulos vastaa edellisen sauman tulosta. Kappaleessa olevan railon ansiosta virtuaalisauma osuu kohdilleen todellisen sauman paikan kanssa. Sauman alku ja loppupään paikka ovat myös poimittu railon kohdalta, joten niiden segmenttien päät ovat linjassa todellisen sauman kanssa.

Kuvassa 14.g) on esimerkki segmentoinnista, missä virtuaalikameran näkemän sauman pintanormaali on melkein suorassa kulmassa kameran havaintovektorin suhteen. Sauman kaikki pisteet eivät rekisteröidy näkyvyysanalyysin yhteydessä, joten osa pattern olioista jää luomatta.

6.3 Kehitysehdotukset

Numeerisen analyysin tekemiseksi segmentoinnin onnistuneisuudesta kuvadatalle tulisi patternien ja todellisen sauman välinen tarkastelu tehdä kuvissa pikselitasolla. Tämän tason tarkastelua ei kuitenkaan ehditty toteuttaa työn puitteissa aikataulullisista syistä. Työssä ei myöskään pystytty tutkimaan kappaleen valmistusvirheistä aiheutuvia poikkeamia.

Segmenttien visualisoinnin jälkeen halutuille pisteille voitaisiin tehdä käyttäjäkohtaiset muutokset jotta patternien sijainti olisi parempi. Siirron pohjalta voitaisiin laskea uudet hitsaussauman sauman kohdekoordinaatit ja käyttää koordinaattien välistä erotusta saumahitsausrobotin radan muuttamiseen.

7. YHTEENVETO

Tässä työssä luotiin ohjelmalle toiminnallisuus, joka segmentoi hitsaussauman automaattisesti mittakuville. Saaduista tuloksista tutkittiin globaalin transformoinnin riittävyyttä hitsaussauman paikan määrittämiseksi. Samalla pyrittiin parantamaan ohjelman käytettävyyttä vähentämällä manuaalisen työn osuutta sen käytössä.

Saadut tulokset osoittavat globaalin transformoinnin olevan riittävä hitsaussauman paikan määrittämisessä, jos sauma hitsataan kappaleen muodon tai railon opastamaan kohtaan. Tietyissä kohdissa sauman paikan ja segmentoinnin heitto voi johtua lämpölaajenemisesta tai kappaleen muodon poikkeamista. Voidaan kuitenkin todeta että työssä käytetyn konenäköjärjestelmän vaatimassa ympäristössä globaali transformointi on riittämätön hitsaussauman paikan määrittämiseksi.

Testisaumoilla automaattisten segmentointien ongelmaksi muodostui niiden alku- ja loppupisteiden paikat. Globaalin transformoinnin ongelmana hitsaussaumoille on nykyisen datan riittämättömyys. Koska hitsaussaumojen pistedata luodaan Siemens NX ohjelman avulla valitsemalla sauman pisteet kappaleen pinnalta ilman opastetta ei voida olettaa että transformoinnin tuloksena segmentoidun sauman päillä ja todellisella saumalla samat pisteet. Mittakuvissa nähtävän todellisen sauman alku- ja loppupisteen paikkaa ei pystytä määrittämään automaattisesti ilman hitsausrobotin käyttämää pistedataa tai manuaalista määrittämistä. Tarvitaan siis enemmän dataa, mikäli globaalia transformointia halutaan käyttää sauman määrittämiseksi.

Työn suurimpana osuutena oli ohjelmiston käytön opettelu ja sen muokkaaminen, jonka loppuunsaattaminen muodostui haasteelliseksi aikataulullisista syistä. Vaikka työ saatettiin loppuun aikataulun puitteissa, ei voida puhua täysin toimivasta ohjelmasta sillä lisätyissä osissa on edelleen paljon korjattavaa. Tulevaisuudessa hitsaussaumojen segmentoinnissa tulisi käyttää lokaalimpaa transformointia globaalin tranformoinnin sijaan. Lokaalissa transformoinnissa sauma transformoitaisiin käyttämällä niiden alku- ja loppupisteitä, sekä yhtä reikäpiirrettä. Toisin kuin globaalissa transformoinnissa, lokaalissa transformoinnissa segmentoidun sauman ja todellisen sauman alku- ja loppupisteiden paikat olisivat samat.

Ohjelmoinnissa käytettiin paljon valmiita ohjelmiston osia, joita muokkaamalla uusi toiminnallisuus saatiin nopeammin testattavaksi. Uudella toiminnallisuudella saatiin testattua globaalin transformoinnin toimivuutta tietyille saumoille, joista saatiin vertailukelpoista informaatiota. Saumojen vertailu jäi kuitenkin lopulta visuaaliselle asteelle, koska numeerisen datan saaminen olisi vaatinut erillisen ohjelman luontia.

8. LÄHTEET

[1] Stephen K, Artikkeli: Machine Vision Makes Its Mark on the Automotive Industry, Saatavissa: http://www.controldesign.com/assets/Media/MediaManager/wp_008_cognex_vision_autos.pdf

[2] P. Rönnholm / H. Haggrén, Maa-57.301 Fotogrammetrian yleiskurssi opetusmateriaali 27.9.2005 s. 1, Saatavissa: http://foto.hut.fi/opetus/301/luennot/301_5_2005.pdf

[3] "Introduction to modern photogrammetry" Edward M. Mikhail, James S. Bethel, J. Chris McGlone. 93. 94 112.

[4] H. Haggrén, Maa-57.301 Fotogrammetrian yleiskurssi opetusmateriaali, Saatavissa: http://foto.hut.fi/opetus/301/luennot/7/7.html

[5] Moffitt, F.H. & E.M. Mikhail, 1980. "Photogrammetry." 3rd Ed. Harper & Row, Inc. N.Y., U.S.A.

[6] Tuote-esite, Saatavissa: http://www.mapvision.fi/mapweb/uploads/Quality_Gate_Brochure_en_2015-12-04.pdf

[7] Avoimen lähdekoodin tekstiparseri, Saatavissa: http://www.grinninglizard.com/tinyxml/

[8] H.I. Shafeek, E.S. Gadelmawla, A.A. Abdel-Shafy, I.M. Elewa, Assessment of welding defects for gas pipeline radiographs using computer vision, NDT & E International, Volume 37, Issue 4, June 2004, Pages 291-299, ISSN 0963-8695, http://dx.doi.org/10.1016/j.ndteint.2003.10.003

[9] Paul R. Wolf, Ph.D., Bon A. Dewitt, Ph.D., Benjamin E. Wilkinson, Ph.D, Elements of Photogrammetry with Applications in GIS, Fourth Edition 2014 McGraw-Hill Education, ISBN: 9780071761123.

9. LIITTEET

Liite 1: Hitsaussauman WS1 ja piirrepisteiden koordinaattidata ennen transformointia

Liite 2: Sauman WS1 transformaatioparametrit

LIITE 1.

```
<parameter-file file-type="Feature Definition File" version="1.0">
  <feature id="BOHRUNG_120">
    <feature-type type="string" value="hole" />
    <normal-direction>
      <normal-i type="double" value="0.000000" />
      <normal-j type="double" value="0.000000" />
      <normal-k type="double" value="-1.000000" />
      <visible-from-both-sides type="boolean" value="false" />
    </normal-direction>
    <feature-edge>
      <point>
        <point-x type="double" value="-399.874390" />
        <point-y type="double" value="-452.924119" />
        <point-z type="double" value="1098.539547" />
      </point>
    </feature-edge>
  </feature>
  <feature id="MP_121">
    <feature-type type="string" value="hole" />
    <normal-direction>
      <normal-i type="double" value="0.000000" />
      <normal-j type="double" value="0.000000" />
      <normal-k type="double" value="-1.000000" />
      <visible-from-both-sides type="boolean" value="false" />
    </normal-direction>
```

```
    <feature-edge>
      <point>
        <point-x type="double" value="415.836429" />
        <point-y type="double" value="-453.646940" />
        <point-z type="double" value="1098.672830" />
      </point>
    </feature-edge>
  </feature>
  <feature id="KR_HI_AKV_LI">
    <feature-type type="string" value="hole" />
    <normal-direction>
      <normal-i type="double" value="0.000000" />
      <normal-j type="double" value="0.000000" />
      <normal-k type="double" value="-1.000000" />
      <visible-from-both-sides type="boolean" value="false" />
    </normal-direction>
    <feature-edge>
      <point>
        <point-x type="double" value="-270.404188" />
        <point-y type="double" value="228.815742" />
        <point-z type="double" value="999.806962" />
      </point>
    </feature-edge>
  </feature>
  <feature id="KR_HI_AKV_RE">
    <feature-type type="string" value="hole" />
    <normal-direction>
```

```
      <normal-i type="double" value="0.000000" />
      <normal-j type="double" value="0.000000" />
      <normal-k type="double" value="-1.000000" />
      <visible-from-both-sides type="boolean" value="false" />
    </normal-direction>
    <feature-edge>
      <point>
        <point-x type="double" value="285.543979" />
        <point-y type="double" value="228.709541" />
        <point-z type="double" value="999.790666" />
      </point>
    </feature-edge>
  </feature>
  <feature id="WS_1">
    <feature-type type="string" value="custom" />
    <normal-direction>
      <normal-i type="double" value="0.320357" />
      <normal-j type="double" value="0.307646" />
      <normal-k type="double" value="-0.895950" />
      <visible-from-both-sides type="boolean" value="false" />
    </normal-direction>
    <feature-edge>
      <point>
        <point-x type="double" value="-256.999606" />
        <point-y type="double" value="36.704725" />
        <point-z type="double" value="937.117080" />
      </point>
```

```
<point>
   <point-x type="double" value="-257.456287" />
   <point-y type="double" value="38.231727" />
   <point-z type="double" value="936.566716" />
</point>
<point>
   <point-x type="double" value="-258.046318" />
   <point-y type="double" value="39.703983" />
   <point-z type="double" value="935.994466" />
</point>
<point>
   <point-x type="double" value="-258.765275" />
   <point-y type="double" value="41.110472" />
   <point-z type="double" value="935.404615" />
</point>
<point>
   <point-x type="double" value="-259.770794" />
   <point-y type="double" value="42.669471" />
   <point-z type="double" value="934.692817" />
</point>
<point>
   <point-x type="double" value="-260.937484" />
   <point-y type="double" value="44.106411" />
   <point-z type="double" value="933.970065" />
</point>
<point>
   <point-x type="double" value="-262.099058" />
```

```
    <point-y type="double" value="45.387035" />
    <point-z type="double" value="933.292079" />
</point>
<point>
    <point-x type="double" value="-263.260633" />
    <point-y type="double" value="46.667661" />
    <point-z type="double" value="932.614093" />
</point>
<point>
    <point-x type="double" value="-264.422207" />
    <point-y type="double" value="47.948286" />
    <point-z type="double" value="931.936107" />
</point>
<point>
    <point-x type="double" value="-265.583782" />
    <point-y type="double" value="49.228912" />
    <point-z type="double" value="931.258120" />
</point>
<point>
    <point-x type="double" value="-266.745356" />
    <point-y type="double" value="50.509537" />
    <point-z type="double" value="930.580134" />
</point>
<point>
    <point-x type="double" value="-267.906931" />
    <point-y type="double" value="51.790162" />
    <point-z type="double" value="929.902148" />
```

```
</point>
<point>
   <point-x type="double" value="-269.068505" />
   <point-y type="double" value="53.070788" />
   <point-z type="double" value="929.224162" />
</point>
<point>
   <point-x type="double" value="-270.230079" />
   <point-y type="double" value="54.351413" />
   <point-z type="double" value="928.546176" />
</point>
<point>
   <point-x type="double" value="-271.446967" />
   <point-y type="double" value="55.693021" />
   <point-z type="double" value="927.835905" />
</point>
<point>
   <point-x type="double" value="-272.663854" />
   <point-y type="double" value="57.034628" />
   <point-z type="double" value="927.125633" />
</point>
<point>
   <point-x type="double" value="-273.880742" />
   <point-y type="double" value="58.376235" />
   <point-z type="double" value="926.415362" />
</point>
<point>
```

```
    <point-x type="double" value="-275.097630" />
    <point-y type="double" value="59.717843" />
    <point-z type="double" value="925.705090" />
</point>
<point>
    <point-x type="double" value="-276.314517" />
    <point-y type="double" value="61.059450" />
    <point-z type="double" value="924.994819" />
</point>
<point>
    <point-x type="double" value="-277.531405" />
    <point-y type="double" value="62.401058" />
    <point-z type="double" value="924.284548" />
</point>
<point>
    <point-x type="double" value="-278.748292" />
    <point-y type="double" value="63.742666" />
    <point-z type="double" value="923.574277" />
</point>
<point>
    <point-x type="double" value="-279.965180" />
    <point-y type="double" value="65.084273" />
    <point-z type="double" value="922.864006" />
</point>
<point>
    <point-x type="double" value="-281.182067" />
    <point-y type="double" value="66.425881" />
```

```
    <point-z type="double" value="922.153734" />
</point>
<point>
    <point-x type="double" value="-282.398955" />
    <point-y type="double" value="67.767488" />
    <point-z type="double" value="921.443462" />
</point>
<point>
    <point-x type="double" value="-283.615842" />
    <point-y type="double" value="69.109096" />
    <point-z type="double" value="920.733191" />
</point>
<point>
    <point-x type="double" value="-284.832730" />
    <point-y type="double" value="70.450704" />
    <point-z type="double" value="920.022920" />
</point>
<point>
    <point-x type="double" value="-285.980081" />
    <point-y type="double" value="71.715648" />
    <point-z type="double" value="919.353236" />
</point>
<point>
    <point-x type="double" value="-287.127433" />
    <point-y type="double" value="72.980592" />
    <point-z type="double" value="918.683552" />
</point>
```

```
<point>
   <point-x type="double" value="-288.274784" />
   <point-y type="double" value="74.245536" />
   <point-z type="double" value="918.013867" />
</point>
<point>
   <point-x type="double" value="-289.422135" />
   <point-y type="double" value="75.510480" />
   <point-z type="double" value="917.344183" />
</point>
<point>
   <point-x type="double" value="-290.569486" />
   <point-y type="double" value="76.775424" />
   <point-z type="double" value="916.674499" />
</point>
<point>
   <point-x type="double" value="-292.086650" />
   <point-y type="double" value="78.541094" />
   <point-z type="double" value="915.763172" />
</point>
<point>
   <point-x type="double" value="-293.508009" />
   <point-y type="double" value="80.386930" />
   <point-z type="double" value="914.856247" />
</point>
<point>
   <point-x type="double" value="-294.830010" />
```

```
    <point-y type="double" value="82.308316" />
    <point-z type="double" value="913.955991" />
</point>
<point>
    <point-x type="double" value="-295.527821" />
    <point-y type="double" value="83.419021" />
    <point-z type="double" value="913.454035" />
</point>
<point>
    <point-x type="double" value="-322.751317" />
    <point-y type="double" value="128.226996" />
    <point-z type="double" value="893.461978" />
</point>
<point>
    <point-x type="double" value="-323.573636" />
    <point-y type="double" value="129.578440" />
    <point-z type="double" value="892.871034" />
</point>
<point>
    <point-x type="double" value="-324.400111" />
    <point-y type="double" value="130.933866" />
    <point-z type="double" value="892.295206" />
</point>
<point>
    <point-x type="double" value="-325.230622" />
    <point-y type="double" value="132.293108" />
    <point-z type="double" value="891.734394" />
```

```
</point>
<point>
   <point-x type="double" value="-326.065098" />
   <point-y type="double" value="133.656044" />
   <point-z type="double" value="891.188659" />
</point>
<point>
   <point-x type="double" value="-326.918744" />
   <point-y type="double" value="135.047432" />
   <point-z type="double" value="890.648558" />
</point>
<point>
   <point-x type="double" value="-327.776347" />
   <point-y type="double" value="136.442397" />
   <point-z type="double" value="890.124222" />
</point>
<point>
   <point-x type="double" value="-328.637832" />
   <point-y type="double" value="137.840811" />
   <point-z type="double" value="889.615720" />
</point>
<point>
   <point-x type="double" value="-329.503123" />
   <point-y type="double" value="139.242546" />
   <point-z type="double" value="889.123127" />
</point>
<point>
```

```
    <point-x type="double" value="-330.372141" />
    <point-y type="double" value="140.647472" />
    <point-z type="double" value="888.646519" />
</point>
<point>
    <point-x type="double" value="-331.602451" />
    <point-y type="double" value="142.455332" />
    <point-z type="double" value="888.043227" />
</point>
<point>
    <point-x type="double" value="-332.977061" />
    <point-y type="double" value="144.145904" />
    <point-z type="double" value="887.481405" />
</point>
<point>
    <point-x type="double" value="-334.491119" />
    <point-y type="double" value="145.720010" />
    <point-z type="double" value="886.966912" />
</point>
<point>
    <point-x type="double" value="-336.242834" />
    <point-y type="double" value="147.259270" />
    <point-z type="double" value="886.480260" />
</point>
<point>
    <point-x type="double" value="-338.035163" />
    <point-y type="double" value="148.590118" />
```

```
    <point-z type="double" value="886.082023" />
</point>
<point>
    <point-x type="double" value="-339.920242" />
    <point-y type="double" value="149.774608" />
    <point-z type="double" value="885.755930" />
</point>
<point>
    <point-x type="double" value="-341.892144" />
    <point-y type="double" value="150.816074" />
    <point-z type="double" value="885.503973" />
</point>
<point>
    <point-x type="double" value="-344.063392" />
    <point-y type="double" value="151.762302" />
    <point-z type="double" value="885.320600" />
</point>
<point>
    <point-x type="double" value="-346.184830" />
    <point-y type="double" value="152.508128" />
    <point-z type="double" value="885.227772" />
</point>
<point>
    <point-x type="double" value="-347.510364" />
    <point-y type="double" value="152.892412" />
    <point-z type="double" value="885.209975" />
</point>
```

```
<point>
   <point-x type="double" value="-349.219381" />
   <point-y type="double" value="153.352333" />
   <point-z type="double" value="885.221498" />
</point>
<point>
   <point-x type="double" value="-350.926554" />
   <point-y type="double" value="153.817933" />
   <point-z type="double" value="885.263696" />
</point>
<point>
   <point-x type="double" value="-352.266119" />
   <point-y type="double" value="154.187612" />
   <point-z type="double" value="885.318358" />
</point>
<point>
   <point-x type="double" value="-353.603738" />
   <point-y type="double" value="154.560577" />
   <point-z type="double" value="885.391933" />
</point>
<point>
   <point-x type="double" value="-354.240776" />
   <point-y type="double" value="154.741462" />
   <point-z type="double" value="885.443179" />
</point>
<point>
   <point-x type="double" value="-354.738028" />
```

```
    <point-y type="double" value="154.885943" />
    <point-z type="double" value="885.499519" />
</point>
<point>
    <point-x type="double" value="-355.896389" />
    <point-y type="double" value="155.229180" />
    <point-z type="double" value="885.663878" />
</point>
<point>
    <point-x type="double" value="-357.178677" />
    <point-y type="double" value="155.608218" />
    <point-z type="double" value="885.841250" />
</point>
<point>
    <point-x type="double" value="-358.280025" />
    <point-y type="double" value="155.925949" />
    <point-z type="double" value="885.954725" />
</point>
<point>
    <point-x type="double" value="-359.456660" />
    <point-y type="double" value="156.253279" />
    <point-z type="double" value="886.015735" />
</point>
<point>
    <point-x type="double" value="-366.428802" />
    <point-y type="double" value="158.150219" />
    <point-z type="double" value="886.165282" />
```

```
            </point>
          </feature-edge>
        </feature>
      </parameter-file>
```

LIITE 2.

```
<?xml version="1.0" ?>
<Transformation_parameters>
   <Shifts>
      <shiftX value="-29.6171" />
      <shiftY value="5.42799" />
      <shiftZ value="-10.6453" />
   </Shifts>
   <Rotation_matrix_elements>
      <r11 value="0.998615" />
      <r12 value="-0.0296702" />
      <r13 value="-0.0434504" />
      <r21 value="0.0291541" />
      <r22 value="0.999497" />
      <r23 value="-0.0124638" />
      <r31 value="0.0437983" />
      <r32 value="0.0111797" />
      <r33 value="0.998978" />
   </Rotation_matrix_elements>
   <Scales>
      <scaleX value="1.00073" />
      <scaleY value="1.00073" />
      <scaleZ value="1.00073" />
```

```
    </Scales>
    <Standard_deviation value="1.14931" />
</Transformation_parameters>
```